PROCEEDINGS
CSCSI/SCEIO

CONFERENCE
1982

SASKATOON, SASKATCHEWAN
17, 18, 19 MAY 1982

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

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

PRESENTED IN CONJUNCTION WITH
PRESENTE EN COOPERATION AVEC

Canadian Information Processing Society
The University of Saskatchewan
Simon Fraser University

University of Saskatchewan
Saskatoon, Saskatchewan
17-19 May/Mai 1982

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

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

Bienvenue à Saskatoon. J'espère que vous serez tous satisfaits de la conférence et l'université pendant votre sejour ici. C'est la quatrième Conférence Nationale de la CSCSI/SCEIO, les trois précédentes ayant été tenues à Vancouver en 1976, à Toronto et à Victoria en 1980. J'ai confiance que la nôtre sera la meilleure.

La CSCSI/SCEIO est une organisation de petite envergure tout en étant très dynamique. En plus de tenir une conférence tous les deux ans, nous publions un bulletin et, en général, nous essayons de réunir des chercheurs (Canadien et autres) intéréssés dans le domaine de l'intelligence artificielle. Nous sommes toujours à la recherche de nouveaux membres et nous vous invitons à vous joindre si le coeur vous en dit.

Une conférence comme celle-ci ne se réalise pas du jour au lendemain. Plusieurs personnes doivent travailler d'arrache-pied en vue de sa réalisation et, dans une organisation telle que la CSCSI/SCEIO, ce travail et fait strictement à titre volontaire. J'aimerais particulièrement remercier Nick Cercone pour son excellent effort à la conception du programme et Brian Funt pour sa remarquable contribution à la publication de celui-ci. Mille mercis au comité du programme pour son excellent travail d'évaluation des résumés. Font partie de ce comité: James Allen, Norm Badler, Mike Bauer, Wayne Davis, Mark Fox, Bill Havens, Hector Levesque, Charles Morgan, John Mylpoulos, Zenon Pylyshyn, Reid Smith et Doug Skuce. D'autres personnes ayant aidées à la correspondance, à la préparation des dossiers, à la production des cartes d'identités, à la documentation technique, à l'inscription, à la traduction, etc. comprennent Peta Bates, Dave Goforth, Blake Ward, Marlene Colbourn, et Charles Colbourn de l'université de la Saskatchewan, ainsi que Josie Backhouse, Kathy Bootle et John Gerdes de l'université Simon Fraser. J'aimerais les remercier tous pour leur fructueux efforts. La coopération et l'assistance financière fournies par les départements "Computing Science" des universités Simon Fraser et de Saskatchewan ont été inestimables. Finalement nous en savons gré au Conseil National de la Recherche Scientifique et due génie du Canada, au governement de la Saskatchewan et à la ville de Saskatoon pour leur precieuse contribution.

Gordon McCalla,
Président Général,
Quatrième Conférence Nationale
de la CSCSI/SCEIO

Welcome to Saskatoon. I hope you will also get a chance to see something of the University, the City and the Province while you are here. This is the Fourth National Conference of the CSCSI/SCEIO, the other three having been held in Vancouver in 1976, Toronto in 1978, and Victoria in 1980. I am confident it will be the best one yet.

The CSCSI/SCEIO is a small organisation as organisations go, but a vigorous one. In addition to hosting a biannual conference, we also produce a newsletter and in general attempt to bring together researchers (Canadian and others) who are interested in computational studies of intelligence. We are always looking for new members and would invite you to join us if you wish.

A conference such as this does not just happen. Many people must work long and hard in order to bring it about and, in an organisation such as the CSCSI/SCEIO, this work is strictly volunteer. I would particularly like to thank Nick Cercone for doing a superb job of putting together the programme and Brian Funt for editing the Proceedings. Many thanks to the Programme Committee, who did yeoman service in refereeing the papers; they included James Allen, Norm Badler, Mike Bauer, Wayne Davis, Mark Fox, Bill Havens, Hector Levesque, Charles Morgan, John Mylopoulos, Zenon Pylyshyn, Reid Smith and Doug Skuce. Other people who helped with the correspondence, envelope stuffing, label production, technical writing, registration, language translation, etc., include Peta Bates, Dave Coforth, Blake Ward, Marlene Colbourn, and Charles Colbourn of the University of Saskatchewan and Josie Backhouse, Kathy Bootle, and John Gerdes of Simon Fraser University. I would like to thank all of them for their efforts. The co-operation and financial assistance provided by the Computing Science Department of Simon Fraser University and the Department of Computational Science at the University of Saskatchewan have been invaluable. Finally, the contributions of the National Sciences and Engineering Research Council of Canada, the government of Saskatchewan, and the City of Saskatoon are gratefully acknowledged.

Gordon McCalla,
General Chairman,
Fourth National Conference
of the CSCSI/SCEIO

OFFICERS OF THE FOURTH BIENNIAL CONFERENCE OF THE CSCSI/SCEIO

General Chairman

Gordon McCalla
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N OWO

Program Chairman

Nick Cercone
Department of Computing Science
Simon Fraser University
Burnaby, British Columbia V5A 1S6

Proceedings Editor

Brian Funt
Department of Computing Science
Simon Fraser University
Burnaby, British Columbia V5A 1S6

OFFICERS OF THE CSCSI/SCEIO

President

Alan Mackworth
Department of Computer Science
University of British Columbia
Vancouver, British Columbia V6T 1W5

Vice-President

Steven Zucker
Department of Electrical Engineering
McGill University
Montreal, Quebec H3A 2K6

Treasurer

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

Secretary

Ray Perrault
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A7

Program Committee

James Allen, University of Rochester
Norm Badler, University of Pennsylvania
Mike Bauer, University of W. Ontario
Wayne Davis, University of Alberta
Mark Fox, Carnegie-Mellon University
Bill Havens, University of British Columbia
Hector Levesque, Fairchild R & D
Charles Morgan, University of Victoria
John Mylopoulos, University of Toronto
Zenon Pylyshyn, University of W. Ontario
Reid Smith, Schlumberger-Doll Research
Doug Skuce, University of Ottawa

TABLE OF CONTENTS

--DATA BASES--

PARALLELISM IN PLANNING AND PROBLEM SOLVING: REASONING ABOUT RESOURCES

David E. Wilkins

SRI International Artificial Intelligence Center

ABSTRACT

The implications of allowing parallel actions in a plan or problem solution are discussed. The planning system should take advantage of helpful interactions between parallel branches, must detect harmful interactions, and, if possible, remedy them. This paper describes what is involved in this and presents some new techniques that are implemented in an actual planning system and are useful in seeking solutions to these problems. The most important of these techniques, reasoning about resources, is emphasized and explained.[1]

1. Introduction

This paper discusses problems faced by an automatic planning system concerned with detecting and responding to interactions in parallel branches of a plan. (This could also be thought of as a problem solving system producing problem solutions.) There are three aspects to this situation: recognizing interactions between branches; correcting harmful interactions that keep the plan from accomplishing its overall goal; taking advantage of helpful effects on parallel branches so as not to produce inefficient plans.

We shall consider a domain-independent planning system that provides some formalism for representing the domain in which the planning will be done (as well as the goals to be achieved by the planner). The system also allows for the representation of operators. These are the system's representation of actions that may be performed in the domain or, in the case of hierarchical planners, abstractions of actions that can be performed in the domain. A plan consists of partially ordered goals and actions produced by applying operators to the initial goal. Actions in the plan explicitly list their effects, i.e., they specifically state which relationships in the world model change their truth value after the action is performed. (This is part of what Waldinger [6] has called the STRIPS assumption.) NOAH [3], NONLIN [5], and SIPE [7] are examples of such planners.

In this discussion, parallelism is considered beneficial. (Two segments of a plan are in parallel if the partial ordering of the plan does not specify that one segment must be done before the other.) In much automatic programming research the goal is to produce a sequential program, so it is often easier to completely order the plan (program) rather than reason about parallel interactions. However, in multieffector environments parallelism is preferable. For example, if there are two robot arms, plans that use them in parallel to accomplish the goal are often preferred to a sequential plan that could get by with only one arm (even though it might take twice as long to accomplish). Our approach, therefore, is to keep as much parallelism as possible and to reason about the interactions that result from it.

[1] The research reported here is supported by Air Force Office of Scientific Research Contract F49620-79-C-0188.

2. Parallel Interactions

The canonical simple problem for thinking about parallel interactions in this context is the three-blocks problem. Blocks A, B, and C are on a table or on one another (only one block may be on another at any one time). The goal is to achieve (ON A B) in conjunction with (ON B C), thus making a three-block tower. (Initially the two goals are represented as being in parallel.) To move the blocks there is a PUTON operator (expressed in the formalism of whatever planning system we wish to talk about) that puts OBJECT1 on OBJECT2. Its definition specifies the goals of making both OBJECT1 and OBJECT2 clear before performing a primitive move action. (The table is assumed always to be clear and a block is clear only when no block is on top of it.) This problem will be used below to provide examples of interactions.

If two branches of a plan are in parallel, an interaction is defined to occur when a goal that is trying to be achieved in one branch (at any level in the hierarchy) is made either true or false by an action in the other branch. Since the actions in a plan explicitly list their effects, it is always possible to recognize such interactions. (In a hierarchical planner, however, they may not appear until lower levels of the hierarchy of both branches have been planned.) By requiring that a goal be involved in the interaction, we attempt to eliminate interactions that we do not care about. For this to succeed, the domain must be encoded so that all important relationships are represented as goals at some level. This will be discussed later.

The planner can possibly take advantage of a situation in which a goal in one branch is made true in another branch (a helpful interaction). Suppose we solve the three-blocks problem, starting with A and C on the table and B on A. In solving the (ON B C) parallel branch, the planner will plan to move B onto C, thus making A clear and C not clear. Now, while an attempt is made to move A onto B in the (ON A B) branch, the goal of making A clear becomes part of the plan. Since A is not clear in the initial state, the planner may decide to make it true by moving B from A to the table (after which it will move A onto B). In this case it would be better to recognize the helpful effect of making A clear, which happens in the parallel branch. Then the planner could decide to do (ON B C) first, in which case both A and B become clear and the (ON A B) goal is easily accomplished later.

The planner must decide whether or not to add more ordering constraints to the plan to take advantage of such effects on a parallel branch. Ordering the parallel branches sequentially is the best solution to this problem because (ON B C) must be done first in any case, but in other problems an ordering suggested to take advantage of helpful effects may be the wrong thing to do from the standpoint of eventually achieving the overall goal. In general, the planner cannot make such an ordering decision without error unless it completely investigates all the consequences of such a decision. Since this is not always practical or desirable, planning systems use heuristics to make such decisions.

If an interaction is detected that makes a goal false in a parallel branch, there is a problematic (i.e., possibly harmful) interaction which may mean that the plan is not a valid solution. For example, suppose the planner does not recognize the helpful interaction in our problem and proceeds to plan to put B on the table and A on B in the (ON A B) branch. The plan is no longer a valid solution (if it is assumed that one of the two parallel branches will be executed before the other). The planner must recognize this by detecting the problematic interaction. Namely, the goal of having B clear in the (ON B C) branch is made false in the (ON A B) branch when A is put onto B. The planner must then decide how to rectify this situation.

As with helpful interactions, there is no easy way to solve harmful interactions. Here too a correct solution may require that all future consequences of an ordering decision be explored. Stratagems other than ordering may be necessary to solve the problem. For example, a new operator may perhaps need to be applied at a higher level. Consider the problem of switching the values of the two registers in a two-register machine. Applying the register-to-register move operator creates a harmful interaction that no ordering can solve, since a value is destroyed. The solution to this interaction involves applying a register-to-memory move operator at a high level in order to store one of the values temporarily. Correcting many types of harmful interactions efficiently seems very difficult in a domain independent planner – domain specific heuristics may be required.

3. Summary of SIPE System

The problem of parallel interactions has been studied at SRI International in the context of a domain-independent planning system. This section briefly summarizes the system and the following section explains how parallel interactions are handled. Ann Robinson and I have designed and implemented a system, SIPE (System for Interactive Planning and Execution monitoring) [7], that supports both interactive and automatic planning. At present, its automatic search is not very knowledgeable. SIPE is designed to allow interaction with users who are able to watch (graphically) and, when desired, guide and/or control the planning process, thus enabling the solution of much more difficult problems than would be possible with the automatic search. The system can keep many alternative plans in readiness, each with the appropriate context, thus making breadth-first or best-first searches easy to implement.

The system provides for representation of domain objects and their invariant properties in a type hierarchy with inheritance. Relationships among these objects that may change over time are represented as predicates in first-order logic, with universal quantification allowed. Predicates are used to describe the preconditions and effects of operators.

The system permits the posting of many types of constraints on the values of variables in a plan (e.g., specifying that the eventual instantiation of the variable must have a certain value for a certain attribute, that it must be the same as the instantiation of another variable, etc.). This allows partial descriptions of objects to be built up so that the planner can accumulate information before making decisions. Constraints help encode many things that would be hard to represent in the predicate calculus language used for preconditions and effects. Constraints are also used to represent information about parallel interactions (see the next section).

The planning is hierarchical; each goal and action can be expanded into a more detailed multistep plan until the primitive level is reached. Plans are represented as procedural nets (similar to those in NOAH [3]), their partial ordering being encoded by successor links in the net. SPLIT and JOIN nodes in the net allow for parallelism in plans.

Operators represent actions in the domain or abstractions of actions (that will eventually be expanded at lower levels in the hierarchy to actual actions). Operators can introduce new variables, impose constraints upon new or old variables, and represent their instructions for expanding a node in a formalism

similar to procedural nets-namely, nodes with slots that are partially ordered by the links between them (see [3]). The slots of a node will represent (among other properties), the name of an action to be performed for an action node, the predicates to be achieved for a goal node, the arguments to be used, and the effects of performing the given action. SIPE allows arguments of an action or goal node to be specified as "resources". As we shall see later, this is very useful for reasoning about parallel interactions.

The effects of actions must be appropriately defined in the operators so the system can use the STRIPS assumption to determine the state of the world in the middle of the plan. Predicates in the effects may contain universal quantifiers. SIPE also permits specification of deductive operators; tightly controlled deductions are performed automatically by the system. This makes the encoding of operators much simpler, since, in general, only the primary effect must be encoded in an operator and most side effects can then be deduced from the primary effect by using the deductive operators.

Many of the above features of the SIPE planning system are extensions of previous domain-independent planning systems. Among such extensions are the following: development of a perspicuous formalism for encoding descriptions of actions; the use of constraints to partially describe objects; the creation of mechanisms that permit concurrent exploration of alternative plans; the incorporation of heuristics for reasoning about resources; use of quantifiers in the effects of actions and mechanisms that make it possible to perform simple deductions (thereby simplifying operator descriptions).

4. Handling Parallel Interactions in SIPE

The SIPE system has produced correct parallel plans for problems in four different domains (the blocks world, cooking, aircraft operations, and a simple robotics assembly task). This section describes new features and heuristics in the system that aid in handling parallel interactions. These fall into four areas: (1) reasoning about resources, which is the major contribution of SIPE; (2) using constraints to generate correct parallel plans; (3) explicitly representing the purpose of each action and goal to help solve harmful interactions correctly; (4) taking advantage of helpful interactions. Other planners have had some of the features (e.g., NONLIN has features similar to the last two areas mentioned), but SIPE develops these ideas with a different emphasis. Our own emphasis has been to represent the information that must be reasoned about in a way that is natural to humans so that the planner can be easily controlled interactively-thus allowing more difficult problems to be solved.

```
OPERATOR: PUTON
ARGUMENTS: BLOCK1, OBJECT1 IS NOT BLOCK1;
PURPOSE: (ON BLOCK1 OBJECT1);
PLOT:
    PARALLEL
        BRANCH 1:
            GOALS: (CLEARTOP OBJECT1);
            ARGUMENTS: OBJECT1;
        BRANCH 2:
            GOALS: (CLEARTOP BLOCK1);
            ARGUMENTS: BLOCK1;
    END PARALLEL

    PROCESS
    ACTION: PUTON.PRIMITIVE;
    ARGUMENTS: OBJECT1;
    RESOURCES: BLOCK1;
    EFFECTS: (ON BLOCK1 OBJECT1);
END PLOT
END OPERATOR
```

Figure 1
a PUTON operator in SIPE

4.1 Resources

In the following discussion, actual examples will be presented to show how resources help with parallel interactions. Figure 1 shows a PUTON operator written in SIPE. The above example of achieving (ON A B) and (ON B C) as a conjunction shows how resource reasoning is helpful. Figure 2 depicts a plan that might be produced by NOAH or NONLIN (or SIPE without making use of resource reasoning) for this problem. Figure 3 shows a plan from SIPE using resources in the operators.

The formalism for representing operators in SIPE includes a means of specifying that some of the variables associated with an action or goal actually serve as resources for that action or goal. Resources are to be employed during a particular action and then released, just as a saw is used during a cutting action. Reasoning about resources is a common phenomenon. It is a useful way of representing many domains, a natural way for humans to think of problems, and, consequently, an important aid to interaction with the system.

SIPE has specialized knowledge for handling resources; merely specifying that something is a resource causes the system to check on resource availability and on resource conflicts. It is often difficult or awkward to keep track of resources in current planning systems (e.g., [3] [5]). Resource availability in the latter would have to be axiomatized and checked in the preconditions of operators, while resource conflicts would have to be caught by the normal problematic interaction detector, which is less efficient (as we shall see below).

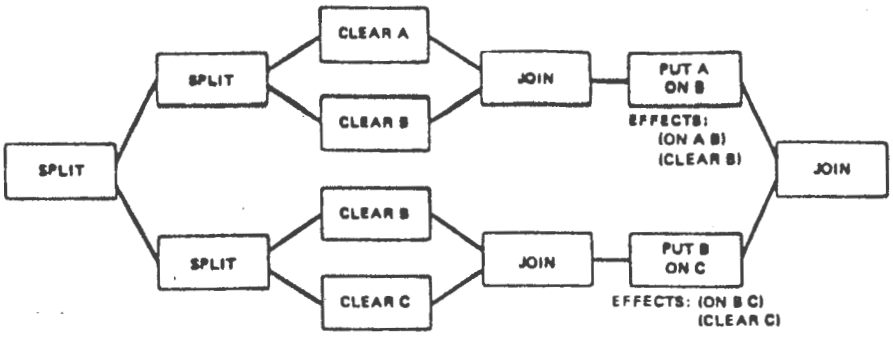


Figure 2
a plan without resources

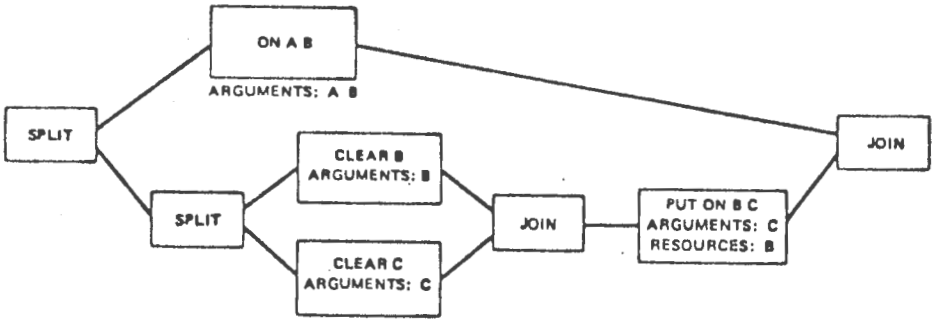


Figure 3
a plan with resources

One advantage of resources, therefore, is that they help in the axiomatization and representation of domains. Declaration of a resource associated with an action connotes that one precondition of the action is the availability of that resource. Mechanisms in the planning system, as they allocate and deallocate resources, automatically ensure that these implicit preconditions will be satisfied. The user of the planning system does not have to axiomatize as a precondition the availability of resources in the domain operators.

Another important advantage of resources is that they help in early detection of problematic interactions on parallel branches. The system does not allow one branch to use an object which is a resource in a parallel branch. In NOAH and NONLIN, both original GOAL nodes are expanded with the PUTON operator or its equivalent. This produces a plan similar to the one shown in Figure 2. The central problem is to be aware that B must be put on C before A is put on B (otherwise B will not be clear when it is to be moved onto C). NOAH and NONLIN both build up a table of multiple effects (TOME) that tabulates every predicate instance asserted or denied in the parallel expansions of the two GOAL nodes. Using this table, the programs detect that B is made clear in the expansion of (ON B C), but is made not clear in the (ON A B) expansion. Both programs then solve this problem by doing (ON B C) first.

SIPE uses its resource heuristic to detect this problem and propose the solution without having to generate a TOME. (SIPE does do a TOME-like analysis to detect interactions that do not fit into the resource reasoning paradigm.) When some object is listed in an action as a resource, the system then prevents that particular object from being mentioned as either a resource or an argument in any action or goal that is in parallel. In the example above, the block being moved is listed as a resource in the primitive PUTON operator because it is being physically moved. Therefore, nothing in a parallel branch should try to move it-or even be dependent on its

current location. This restriction is enforced by not allowing a predicate on a parallel branch to mention the resource. This is a strong restriction (though useful in practice) and can be avoided by using shared resources (discussed below). Thus, as soon as the expansion of (ON B C) with the PUTON operator is accomplished and the plan in Figure 3 produced, SIPE recognizes that the plan is invalid since B is a resource in the expansion of (ON B C) and an argument in (ON A B). This can de detected without expanding the (ON A B) goal at all and without generating a TOME.

Not being able to refer to a resource in another branch is sometimes too strong a restriction. SIPE also permits the specification of shared resources, whereby the same object can be a resource or an argument in a parallel branch if certain conditions for the sharing are met. (Currently shared resources are sharable under all conditions as the sharing conditions have not yet been implemented.)

Resources help in solving harmful interactions, as well as in detecting them. In these interactions no goal is made false on a parallel branch; there is simply a resource conflict. However, if the resource availability requirements were axiomatized with predicates, an availability goal would be made false on a parallel branch. Thus, resource conflicts are considered to be problematic interactions. SIPE uses a heuristic for solving resource-argument conflicts. Such an interaction occurs when a resource in one parallel branch is used as an argument in another parallel branch (distinct from a resource-resource conflict, in which the same object is used as a resource in two parallel branches). This is the type of conflict that occurs in the plan in Figure 3, since B is a resource in the primitive PUTON action and an argument in (ON A B).

SIPE's heuristic for solving a resource-argument conflict is to put the branch using the object as a resource before the parallel branch using the same object as an argument. In this way SIPE decides that (ON B C) must come before (ON A B) in Figure 3. This is done without generating a TOME, without expanding both nodes, and without analyzing the interaction. The assumption is that an object used as a resource will have its state or location changed by such use; consequently, the associated action must be done first to ensure that it will be "in place" when later actions occur that use it as an argument. This heuristic is not guaranteed to be correct, but it has proven useful in the four domains encoded in SIPE. (The user can easily prevent the employment of this heuristic by interacting with the system.)

To take full advantage of resources, the system posts constraints. This capability is discussed briefly in the next section.

4.2 Constraints

Unbound variables in a plan can accumulate various constraints in SIPE, thus building up a partial description of the object. (Constraints were first used in planning by Stefik in his domain-specific planner MOLGEN [4].) This is useful for taking full advantage of resources to avoid harmful interactions. When variables that are not fully instantiated are listed as resources, the system posts constraints on the variables which point to other variables that are potential resource conflicts. When allocating resources, the system then attempts to instantiate variables so that no resource conflicts will occur. For example, if a robot arm is used as a resource in the block-moving operators, the system will try to use different robot arms (if they are available) on parallel branches, thus avoiding resource conflicts. If only one arm is available, it will be assigned to both parallel branches in the hope that the plan can later be ordered to resolve the conflict. In this way many harmful interactions are averted by intelligent assignment of resources.

4.3 Solving Harmful Interactions

The difficulty entailed in eliminating harmful interactions has already been discussed. However, if the system knows why each part of the plan is present, it can use this information to come up with reasonable solutions to some harmful interactions. Suppose a particular predicate is made false at some node on one parallel branch and true at another node on another parallel branch. Depending on the rationale for including these nodes in the plan, it may be the case that the predicate is not relevant to the plan (an extraneous side effect), or must be kept permanently true (the purpose of the plan), or must be kept only temporarily true (a precondition for later achievement of a purpose).

Solutions to a harmful interaction may depend on which of these cases hold. Let us call the three cases side effect, purpose, and precondition, respectively, and analyze the consequent possibilities. If the predicate in conflict on one branch is a precondition, one possible solution is to further order the plan, first doing the part of the plan from the precondition on through its corresponding purpose. Once this purpose has been accomplished, there will be no problem in negating the precondition later. This solution applies no matter which of the three cases applies to the predicate in the other conflicting branch.

In case both conflicting predicates are side effects, it is immaterial to us if the truth value of the predicate changes and thus no real conflict exists. In the case of a side effect that conflicts with a purpose, one solution is to order the plan so that the side effect occurs before the purpose; thus, once

the purpose has been accomplished it will remain true. When both conflicting predicates are purposes, there is no possible ordering that will achieve both purposes at the end of the plan. The planner must use a different operator at a higher level or plan to reachieve one of the purposes later. However, none of the above suggestions for dealing with interactions can be guaranteed to produce the best solution.

This has been a brief summary of SIPE's algorithm for dealing with problematic interactions. Systems like NOAH and NONLIN do similar things. However, SIPE provides methods for more precise and efficient detection. It should be emphasized that many interactions that would be problematical in the other systems are dealt with in SIPE by the resource-reasoning mechanisms and therefore do not need to be analyzed. When interactions are being analyzed, SIPE requires that one of the conflicting predicates be a goal (not just a side effect) at some level in the hierarchy. In this way, interactions between side effects that pose no problems are not even detected. This requires that all important predicates be recognized as goals at some level, which is easily done in SIPE's hierarchical planning scheme. The system also distinguishes between main and side effects at each node in the plan. This makes it easy to tell which predicates are of interest to us at any level of the plan without looking up the hierarchy (since higher-level goals will become main effects at lower-level actions).

SIPE also provides for exact expression of the purpose of any goal in its operators. NOAH used a heuristic, according to which the last node in an expansion was the purpose of the expansion. This is not always accurate and in SIPE a node can specify any later node in the expansion as its purpose. This enables better analysis of problematic conflicts.

4.4 Changing Goals to Phantoms Through Linearization

SIPE recognizes helpful interactions and will try to further order the plan to take advantage of them, although the user can control this interactively if he wishes. If a goal that must be made true on one parallel branch is actually made true on another parallel branch, the system will order the plan so that the other branch occurs first (if this causes no other conflicts). The goal can then be changed to a phantom and need not be achieved.

NOAH was not able to take advantage of such helpful effects. NONLIN did have an ability to order the plan in this way. This is an important ability in many real-world domains, since helpful side effects occur frequently. For example, if parallel actions in a robot world both require the same tool, only one branch need plan to get the tool out of the tool box; the other branch should be able to recognize that the tool is already out on the table.

5. Related Work

Much planning-like research does not fit into the context we have defined here–because it is specialized to one domain, because it does not make the STRIPS assumption, or because it does not reason about parallel actions. For example, much automatic programming research does not deal with parallel actions or does not make the STRIPS assumption. The STRIPS planner [1] itself does not deal with parallel actions.

The most relevant systems, NOAH and NONLIN, have already been compared with SIPE throughout this paper. Both NOAH and NONLIN find interactions by generating a TOME (table of multiple effects). The TOME finds all interactions, even harmless ones (i.e., those in which both predicates are side effects), work which SIPE avoids. SIPE also provides for explicit designation of purpose for preconditions that NOAH does not provide. NONLIN provides a similar capability with its "goal structure". The most significant improvement in SIPE is the use of resource reasoning (and the ability to post constraints), which averts many harmful interactions and enables many others to be recognized quickly and solved. Neither NOAH nor NONLIN provides a similar capability.

6. Conclusion

We have defined the problem of parallel interactions in a context that is not unique to SIPE. The difficulty of solving harmful interactions was discussed and a case-by-case analysis of different situations presented. An actual planning system was described that incorporates several new mechanisms able to assist in dealing with the parallel interaction problem. The most significant of these mechanisms is the ability to reason about resources. Combined with the system's ability to post constraints, resource reasoning helps the system avoid many harmful interactions, helps it recognize sooner those interactions that do occur, and helps the system solve some of these interactions more quickly.

REFERENCES

1. Fikes, R., Hart, P., and Nilsson, N., "Learning and Executing Generalized Robot Plans," *Readings in Artificial Intelligence*, Nils Nilsson and Bonnie Webber, ed., Tioga Publishing, Palo Alto, California (1981), pp. 231–249.

2. Robinson, A.E., and Wilkins D.E., "Representing Knowledge in an Interactive Planner," *Proceedings of the First Annual Conference of the AAAI*, Stanford, California (August 1980), pp. 148–150.

3. Sacerdoti, E., *A Structure for Plans and Behavior*, Elsevier, North-Holland, New York, 1977.

4. Stefik, M., "Planning With Constraints," *Report STAN-CS-80-784*, Computer Science Department, Stanford University, Ph.D. Dissertation (1980).

5. Tate, A., "Generating Project Networks," *Proceedings IJCAI-77*, Cambridge, Massachusetts (August 1977), pp. 888–893.

6. Waldinger, R., "Achieving Several Goals Simultaneously," *Readings in Artificial Intelligence*, Nils Nilsson and Bonnie Webber, ed., Tioga Publishing, Palo Alto, California (1981), pp. 250–271.

7. Wilkins, D.E., and Robinson, A.E., "An Interactive Planning System," *Artificial Intelligence Center Technical Note 245*, SRI International, Menlo Park, California (March 1980).

INCREMENTAL PLANNING IN A PROBABILISTIC MODEL FOR UNCERTAIN PROBLEM SOLVING

Arthur M. Farley*
Computer and Information Science Department
University of Oregon
Eugene, Oregon 97403

Abstract

A probabilistic model of problem solving under conditions of uncertainty is described. Information gathering operators are introduced to make feasible the solution of reasonably constrained, uncertain problems. The notion of relaxed solution plans is defined, based upon the expected degree of goal space satisfaction. Our model leads to a straightforward scheme for incremental planning.

I. Introduction

Problem solving refers to activity undertaken to reduce and eventually eliminate differences between current and desired (or goal) situations. Three basic types of problem-solving activity can be distinguished: problem representation, solution determination (planning), and plan execution. Traditional artificial intelligence (AI) approaches to problem solving tended to consider these activities as successive phases of the problem-solving process. This proved appropriate for well-structured, puzzle-like problems. Recently, increased attention has focused on applying AI solution techniques to problems arising in real-world contexts. Such contexts confront an agent with uncertainty due to a variety of factors, including imperfect interpretation of environmental information, unreliable execution of planned actions, and unpredictable interaction with other agents or natural systems. The uncertainty inherent in real-world problem solving motivates considerations of more complex control among the three basic activity types.

Incremental planning refers to the interleaving of plan execution with planning. In this paper, we define a scheme for incremental planning within a probabilistic model for uncertain problem solving. The scheme maintains a minimum expected probability of plan success, replanning when insufficiently solved contingencies are encountered during plan execution.

* Research performed while author on leave at the Artificial Intelligence Center, SRI International, Menlo Park, California 94025, and partially supported by ONR Contract No. N00014-81-C-0115.

II. A Probabilistic Model for Uncertain Problem Solving

Our probabilistic model is a straightforward generalization of the well-studied notion of problem space. A problem space is a four-tuple (SS,OP,cs,GS), where SS is the state space, a set of allowable state descriptions; OP is a set of operators, functions mapping state descriptions to state descriptions; cs is the current (or initial) state description; and GS is the goal space, a subset of SS. Problem representation activities generate a problem space representing a particular problem; planning is carried out within the resultant problem space. Heuristic methods to guide solution search have been long-standing topics of AI research. A heuristic search method is admissible if it is guaranteed to determine a minimum-cost solution to a problem, if a solution exists. Admissible heuristic search algorithms have been characterized in terms of required properties of associated evaluation functions [4].

An uncertain problem is represented in terms of an uncertain problem space. We define an uncertain problem space to be a six-tuple (SS,USS,UOP,ucs,GT,ST). SS and GS are as in the certain case. USS is the space of uncertain states; ucs is the uncertain current state. An uncertain state is a set of component state descriptions from SS, with each component having an associated probability such that the sum of component probabilities is equal to 1.0. UOP is a set of unreliable operators, functions mapping uncertain states to uncertain states. In [2], we discuss the specification of unreliable operators as Markov processes over state descriptions. Applying such an operator to an uncertain state us yields the probabilistically weighted combination of results over components of us. Our uncertain problem space differs from the models described for most Markov control problems that assume stochastic operators but certain (i.e., completely observed, singleton) states [1,4].

ST is the solution threshold, $0.0 \leq ST \leq 1.0$. With each uncertain state us we associate a degree of goal space satisfaction, dsat(us), equaling the sum of the probabilities of components of us that are also in GS. We initially define a solution to an uncertain problem

to be a sequence of unreliable operators uol,..., uok such that uok(...uol(ucs)...) = ugs with dsat(ugs) ≥ ST. The notion is that a solution transforms an uncertain current state into one containing components that satisfy goal space criteria with combined probability greater than or equal to the solution threshold ST. A forward-directed search algorithm can be readily defined, halting when it selects a state us for development such that dsat(us) ≥ ST. In [2] we discuss the specification of heuristic evaluation functions that result in admissible search algorithms (in terms of expected cost) for uncertain problem solving.

Our model as defined thus far would have difficulty yielding solutions to uncertain problems having normally constrained goal spaces and reasonably high solution thresholds. Application of an unreliable operator can be expected to result in an uncertain state containing more components with lower probabilities than that to which the operator is applied. Furthermore, an operator may result in goal-directed states for some components and not for others; another operator may produce complementary results. The needs to control state disunity and to realize pragmatic focusing prompt the addition of information gathering operators (IGOPs) to our model. An IGOP obtains information as to the presently existing environmental situation when performed during plan execution. Application of an IGOP at planning time distributes one uncertain state into several uncertain states, each containing the subset of components consistent with a possible IGOP result. We assume that the possible results of an IGOP serve to partition SS. Each result has a probability equal to the sum of probabilities of the components consistent with it. Plans containing IGOPs become rooted trees with uncertain states as nodes, unreliable operators labeling arcs, and IGOPs labelling arc sets. Branching occurs when an IGOP is applied. The uncertain current state ucs is the root of such plan trees.

Our original definition of uncertain problem solution must be extended to cover plans containing IGOPs. A strict generalization is that a solution is a plan tree with leaf set LF such that dsat(lf) ≥ ST for each leaf lf in LF. This definition guarantees that the final uncertain state produced by any plan execution satisfies the condition of our original solution definition. However, it requires that a sufficient plan be generated for every contingency without consideration of likelihood. With each uncertain state us of a plan, we can associate the probability uprob(us) of its being encountered during plan execution, equaling the product of probabilities of IGOP results on the path between us and ucs (or 1.0 if none have been applied).

Let us now 'relax' our original solution definition to reflect the expected degree of goal satisfaction: a solution is a plan with leaf set LF such that SUM [uprob(lf)*dsat(lf)] ≥ ST, summed over all lf in LF. We can associate with each uncertain state us of the plan the value past(us) equal to the expected degree of satisfaction realized by the subplan having us as root. A relaxed solution plan is such that psat(ucs) ≥ ST. This relaxed solution definition allows a plan to ignore unlikely contingencies when others have been solved to sufficient degree.

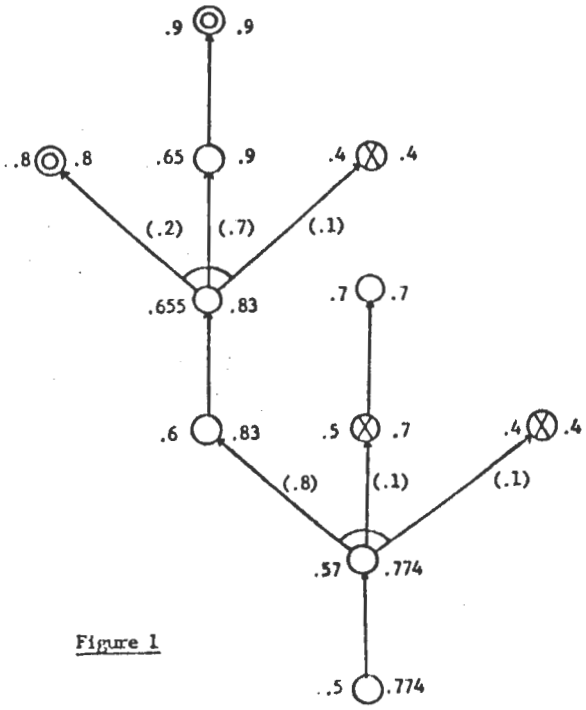


Figure 1

Figure 1 presents a schematic solution plan for an uncertain problem, illustrating some of the notions introduced above. The root state ucs is at the bottom. IGOPs are represented by multiple arcs leaving a state; the probability of each IGOP result is indicated in parentheses. Each node represents an uncertain state us. The value of dsat(us) is indicated to the left of a node; that of psat(us) is to the right. The plan shown could be a relaxed solution to an uncertain problem with ST = .77. If the strict solution definition were required, ST could be at most the minimum of leaf satisfactions (or .40). Issues of determining relaxed solution plans in conjunction with the use of (perfect and imperfect) IGOPs are discussed in [2].

We want to define a scheme that interleaves planning with plan execution, requiring only relaxed solution plans yet guaranteeing satisfaction of the desirable property of our original solution definition, that the uncertain state us reached by plan execution is such that dsat(us) > ST.

III. Incremental Planning

Let us assume the existence of a planning system for uncertain problems, activated by calling PLAN(ucs,GS,ST,UOP,IGOP) with corresponding arguments from an uncertain problem space; IGOP is a set of available information gathering operators. PLAN returns a plan with ucs as root satisfying the relaxed solution definition given above. We also assume the existence of a plan execution system that is activated by calling EXECUTE(ustate, plan) which executes the operator from ustate (unreliable or IGOP) that is indicated in plan, returning as its value the resultant uncertain state in plan.

Our incremental planning scheme is presented below as procedure INCREMENTAL:

```
procedure INCREMENTAL(ucs,GS,ST,UOP,IGOP)
   begin
     ustate <- ucs;
     current plan <- PLAN(ustate,GS,ST,UOP,IGOP);
     While  [ dsat(ustate) < ST ]  do
       begin
         ustate  - EXECUTE(ustate,current plan);
         If  [psat(ustate) < ST ] then
          current plan <- PLAN(ustate,GS,ST,UOP,IGOP)
       end
    end.
```

INCREMENTAL replans whenever it encounters an uncertain state us during plan execution such that psat(us) < ST. In Figure 1, those states marked by an 'x' would trigger replanning, assuming ST = .77. The scheme guarantees that current plan always satisfied our relaxed solution definition and that, when INCREMENTAL halts, our original solution condition is met. In Figure 1, those states marked by an 'o' correspond to possible final states. Actually, PLAN need not always find a plan such that psat(ustate) ≥ ST. If the state disunity of ustate were particularly high, PLAN could just as well propose execution of a single IGOP, forestalling search for a relaxed solution plan until more were known as to presently existing conditions.

IV. Example

In this section, we demonstrate application of our notions by considering a simplified robotics environment. In this real-world context, a robot has the task of acquiring and then attaching screws to an object that is being assembled. The robot can pick up a screw from a parts container and screw it into a pre-drilled hole. We model the possible robot actions by the following four operators, two unreliable, one reliable, and one an imperfect IGOP:

UOP	OUTCOME	P(OUTCOME)
GET-SCREW	HAVE(GOOD(SCREW))	.8
(G-S)	HAVE(BAD(SCREW))	.2
PRE: NOT(HAVE(SCREW))		
IN-SCREW	IN(SCREW)	.9
(I-S)	NOT(HAVE(SCREW))	.1
PRE: HAVE(SCREW)		

OP	OUTCOME
DROP-SCREW	NOT(HAVE(SCREW))
(D-S)	(reliable)
PRE: HAVE(SCREW)	

IGOP	ACTUAL	RESULT	P(RESULT)
TEST-SCREW	GOOD(SCREW)	GOOD	1.0
(T-S)	BAD(SCREW)	BAD	.95
PRE: HAVE(SCREW)		GOOD	.05

The GET-SCREW operator represents the notion that only 80% of the screws are good; we assume they are acquired from a container that is refilled frequently. The operator IN-SCREW indicates the success that the robot has in attaching a screw that it has. We assume that if the robot fails, the screw falls to the floor and is swept away. Similarly, if the robot drops a screw, it is removed from the context (i.e., work station). Note that our operator definitions are incomplete, conveying only main effects. The IGOP TEST-SCREW can be used by the robot to evaluate a screw it has acquired. The operator is imperfect, categorizing a bad screw as good occasionally.

Consider the simplest plan for putting in a screw, as shown in Figure 2. HAVE, BAD, GOOD, SCREW, IN, and NOT are abbreviated by H,B,G,S, I and N, respectively. This plan solves the problem of having a screw in a hole with probability .90, the limit of IN-SCREW. However, if we are interested in having a good screw in a hole, our solution threshold could at most be .72, for this plan to be considered a solution. The plan of Figure 3 improves upon this solution threshold.

Figure 2

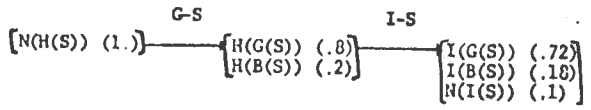
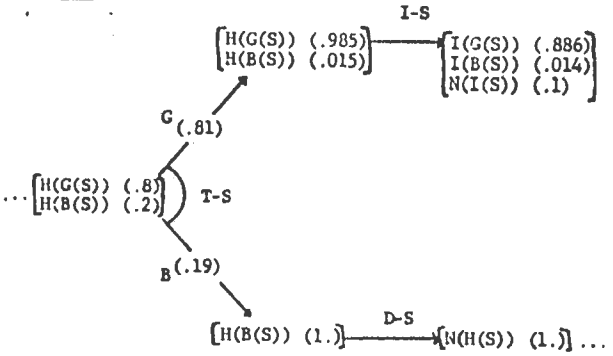


Figure 3



In the plan of Figure 3, if the robot sees that it has a bad screw, it drops the screw and then applies the plan of Figure 2. If the robot believes it has a good screw, it continues by attaching that screw. Recall that a few bad screws can slip by the inspection. This plan satisfied the goal of having a good screw in a hole with probability ≈ .854 (i.e., .81 * .885 + .19 * .72). Note that the probability of having either type of screw in remains at .90, as limited by the reliability of the IN-SCREW operation. To improve upon this, we would need to add an IGOP that can assess whether a screw has gone in or not. Our model can be embellished in a variety of ways to better represent relevant aspects of robotic assembly tasks. The principles of plan representation and evaluation would remain the same.

Incremental planning could be applied in robotic assembly contexts to allow realization of plans that repeat themselves as subplans. For example, if our robot found that it had a bad screw, it could simply drop the screw and submit to replanning, generating the same plan that it had initially (e.g., by recall). The robot would continue to acquire screws until it believed it had a good one, realizing an upper limit of reliability of .885. Robotic assembly systems that incorporate execution monitoring and adaptive control represent a real-world application of incremental planning that is currently receiving considerable research attention.

V. Discussion

For only one type of context has AI research examined in any detail the interleaving of planning with plan execution. Game playing shares with real-world problem solving the uncertainty due to interaction of multiple agents. Searching a game tree of possible futures before every move is consistent with an incremental planning perspective. Recent plan-based approaches to chess, as represented by PARADISE [5], better reflect our notion of executing an extant plan until an unconsidered contingency arises. Opponent move generation could be modeled as an unreliable operator, adding probabilistic factors to game plan evaluation.

An incremental planning approach to the control of problem solving activities is particularly appropriate to problems for which no complete solution plan can be expected to be found or to problems which place strict time limits on the response time of an agent. Playing a complex game such as chess represents an example of the former class of problems. Modern (electronic) warfare presents problems in which quick reaction must be made in response to possible threats by application of imperfect sensors and unreliable countermeasures. On a more everyday scale, driving a car through a crowded city poses similar problems, suggesting an interaction between a completed abstract plan (a selected route) and incremental, execution planning.

In this paper we have presented a general scheme for incremental planning based upon a probabilistic model of uncertain problem solving. Feldman and Sproull [3] discuss the notion of incremental planning in decision-theoretic terms. In their model, actions are reliable and initial uncertainties are expressed as probabilistic parameters of the environment. Goals, actions, and planning activity are assigned utilities (or costs) and decisions are made to optimize expected utility. Their model could be extended to directly represent uncertain states and include unreliable operators. Our incremental planning scheme could be extended to consider utility as well as degree of goal space satisfaction when making replanning decisions.

Incremental planning is just one of many capabilities that will be required if AI systems are to deal effectively with problem solving in real-world contexts.

References

[1] Bertsekas, D. P., Dynamic Programming and Stochastic Control, New York, New York: Academic Press (1976).

[2] Farley, A. M., "A probabilistic model for uncertain problem solving," Technical Note 256, Artificial Intelligence Center, SRI International, Menlo Park, California (December 1981); submitted for publication.

[3] Feldman, J. A., Dynamic Programming and Markov Processes, New York, New York: Wiley Publishing Company (1960).

[4] Nilsson, N. J., Principles of Artificial Intelligence, Palo Alto, CA: Tioga (1980).

[5] Wilkins, D., "Using patterns and plans in chess," Artificial Intelligence 14, pp. 165-203 (1980).

PLAN FORMATION IN LARGE, REALISTIC DOMAINS

N.S. Sridharan and J.L. Bresina

Laboratory for Computer Science Research, Rutgers University
New Brunswick, New Jersey USA 08903

I Introduction

The problem of plan formation is central to the endeavor of Artificial Intelligence. Despite its importance, the field has still not developed a unified plan generation framework that can be customized to realistic domains – be it planning of machining steps in a varied manufacturing process or routing and scheduling for merchandising, or planning corporate transactions to attain stated objectives. The state of the art is such that each new application requires specialized problem formulation, design of representation, and design of planning procedures and heuristics. The task specific activities often eclipse the realization of a general framework. A scan of the literature reveals a collection of specific problem solving attempts for specific domains [1], [2] and a few explorations of general techniques [3]. There still remains a need for developing an interactive general planning facility which can be customized for different tasks by incorporating task-specific planning methods and task knowledge. It is still a research question how a variety of different strategies may be combined into one

II Context and scope of our research

We have been studying plan generation in everyday settings [4], as an important component of the larger problem of plan recognition [5]. Plan Recognition is the task of arriving at an hypothesis about an actor's plan from observations, i.e. descriptions of action performed by the actor. A "viable" plan hypothesis is required to be self-consistent motivationally, consistent with the observations, and well-supported by the observations. Plan Recognition proceeds by a process we have characterized as "hypothesize and revise". The elicitation of an initial hypothesis can be done in a number of ways – plan generation from a goal attributed to the actor is one of them [6].

From our studies we have gained an understanding of how the human information processing system plans *robustly* in the presence of sparse and unreliable information, and does this under *limitations* of processing resources. In this paper we discuss some of the planning techniques developed to study human planning. We have designed and implemented a system, PLANX10, which consists of a collection of these techniques. The discussion is focused on the *engineering* issue of how a planning system may be organized to work effectively in large and realistic task domains. We include a small example of plan generation. We have not carried out a large application yet, but this report marks the beginning of our concern with this important engineering question.

III Other recent approaches

From an engineering point of view, several research efforts have developed techniques and facilities that directly address or indirectly bear on aspects of effective planning in large and realistic task domains. We mention some of the recent approaches to: (a) generating and manipulating large and complex plans; (b) handling large and complex knowledge bases; (c) operating with an incomplete world model and/or knowledge base.

Approaches for dealing with large and complex plans include: (i) use of abstraction in planning [7], [8], [9], [10]; (ii) use of plans with actions only partially ordered over time [11]; (iii) incremental planning and plan repair [12], [9], [7], [13]; (iv) consideration of alternate plans [14], [7]; (v) opportunistic planning [15]; (vi) distributed planning [16], [12]; (vii) use of explicit resource-declarations with actions [7]; (viii) accounting for prepatory and cleanup actions [7]; and (ix) the postponement of parameter selection for actions until appropriate constraints have been developed [2], [7].

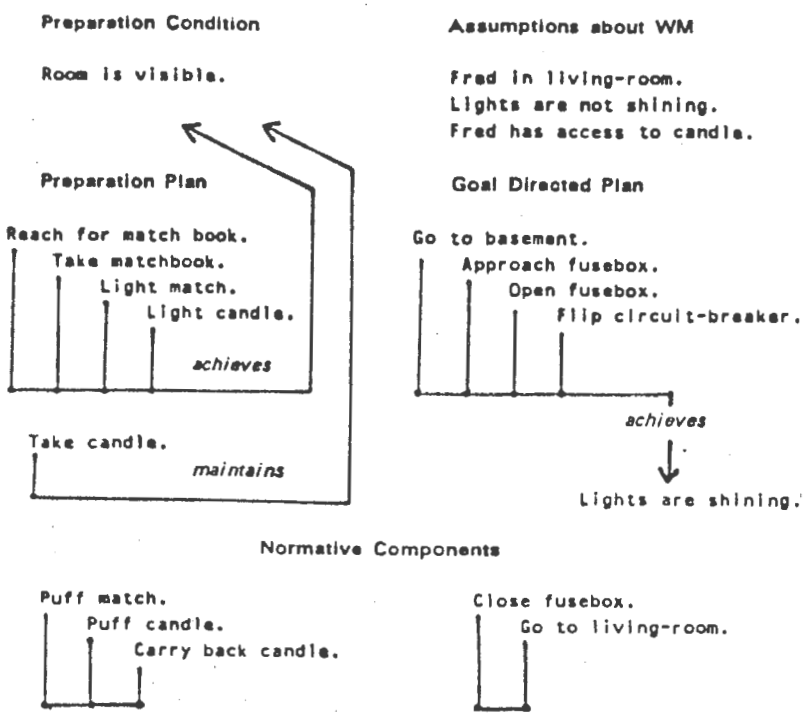
Approaches for handling large and complex knowledge bases include: (i) using a distributed system of multiple knowledge sources and resource limited information retrieval [17]; and (ii) enrichment of the language of interaction with the knowledge base [18].

One approach to cope with missing knowledge is to allow assistance from an interactive user [7]. A central issue to working with incomplete knowledge involves the use of partial descriptions and object descriptions which have no current referent in the world model; i.e., *phantom objects* [7], [2], [14].

IV Components of common-sense plans

A complex task may involve the construction of plans consisting of a large number of actions, may have a complex structure to the solution and may require a diverse variety of knowledge in the task domain. Furthermore, the knowledge will most likely be incomplete and perhaps inconsistent. Typically, current plan generators formulate plans consisting of 3-20 actions. Large tasks may require plans that include 10-200 actions. The principles of Artificial Intelligence [19] offer many useful engineering ideas for coping with this kind of search complexity. Rather than focus on search, our study has concentrated on the structural complexity of commonsense plans. We have decomposed a plan into three structural parts: the familiar goal-directed component, a prepatory component, and a normative actions component. Each component includes statements of goals to achieve as well as goals to maintain.

Figure 1: Components of an example commonsense plan



Consider the example in Figure 1. The primary component contains a goal and a set of subgoals forming a partial order. The final action will have as goal the goal of the entire plan, e.g. lights are shining. The goal-directed actions require visibility of the local area as a precondition and the preparation plan achieves this by lighting a candle and then adding other actions to maintain this visibility, i.e. by carrying the candle to the basement. Note that the preparation condition is not a subgoal for any plan unit in the primary plan but is a condition that needs to be achieved and maintained throughout. The normative component includes actions which do not contribute to the main goal, rather, these actions arise from consideration of the norms (e.g. politeness) and conventions governing the setting (e.g. concerning safety and economy) and roles of the actors (e.g. customer, guest).

We have rejected the design option of extending each action with preparation conditions and normative rules (as in [7]) in favor of viewing the whole plan as being dissected into components. By adopting this strategy we seek to reduce the structural complexity of the overall plan and to control the complexity of the reasoning processes involved in planning. In the example plan, the same preparation condition, visibility, is needed for every action in the goal-directed component. It is more economical to let this condition be planned for just once. If actions in the goal-directed component (e.g. going to the basement) falsify the preparation condition, actions may be added to maintain it (e.g. take the candle). Further, in a planning situation where the construction of the goal-directed component requires backtracking search, our design strategy pays off. The preparation actions are not involved during this stage, thus search is simplified. Note that the actions in the normative component are temporally constrained only weakly in relation to the actions in the main parts of the plan.

V Types of knowledge

Realistic task domains require diverse *types of knowledge* in large amounts. There is a need to ensure that increased knowledge is not a liability to the planning process. In realistic task domains we cannot make the usual assumptions about the "correctness" and "completeness" of knowledge presented to the program. We table the issue of correctness of the knowledge base, for now, and elaborate the consequences of having incomplete knowledge. There are two aspects to the incompleteness: incompleteness in the specification of the problem and incompleteness in the action knowledge. Discussion can be simplified if we assume that the statement of the goal to achieve/prevent/maintain is clear and complete; but that it is difficult for us to provide a complete description of the initial situation that includes *all relevant* information. Even if such completeness is possible in principle, to insure it in large scale situations would be tedious and thus error-prone. Consequently, planning algorithms must be designed to make appropriate and needed *default assumptions* about the initial situation by appeal to general knowledge. These assumptions should be explicit so that they can be evaluated as part of selecting the solution from candidates.

Figure 2: Examples of Rules

Precondition
[Given an action, retrieve its preconditions.]
```
((POUR X (AGENT (PERSON P)) (FROM (CONTAINER C)))
 T
 (PERSON P (INHAND (CONTAINER C))))
```
In order for a person to pour from a container, s/he must have it in hand.

Act-Select :
[Given a goal, retrieve the actions that achieve it.]
```
((OBJECT M (ACCESSIBLE (PERSON P)))
 ((OBJECT M (WITHIN (CONTAINER C))))
 (OPEN X (OF (CONTAINER C)) (AGENT (PERSON P))))
```
To gain access to an object that is within a container, open the container.

Normally-True :
[Given an object description, determine if it is normally true.]
```
((GLASS G (CLEAN YES))
 ((GLASS G (WITHIN (CONTAINER C (TYPE CABINET)))))
 T)
```
It is normally true that a glass in a cabinet is clean.

Outcome :
[Given an action, retrieve its outcomes.]
```
((OPEN X (AGENT (PERSON P)) (OF (CONTAINER C)))
 ((CONTAINER C (CONTAINS (OBJECT O))))
 (OBJECT O (ACCESSIBLE (PERSON P))))
```
When a person opens a container, any object within the container becomes accessible to him/her.

Act-Customize :
[Given an action, retrieve possible action refinements.]
```
((POUR X (INTO (GLASS G)) (OF WATER))
 ((BOTTLE B (TYPE WATER-BOTTLE)))
 (POUR X (FROM (BOTTLE B))))
```
To pour water into a glass from a water bottle, pour from a water bottle.

Normally-False :
[Given an object description, determine if it is normally false.]
```
((CONTAINER B (EMPTY YES))
 ((CONTAINER B (WITHIN (CONTAINER R (TYPE REF)))))
 F)
```
It is normally-false that a container in a refrigerator is empty.

The use of default reasoning then leads us to formulate explicit annotations to be made of key planning decisions. All knowledge used by the plan generator is in the form of *Rule Sets*, each with rules of a uniform declarative format. Some of these rule sets are: *Preconditions, Outcomes, Act-Select, Act-Customize, Normally-False, Normally-True* [see Figure 2]. A rule consists of three parts – the first part is the *Applicability Test*, the second part is the *World Model Test*, and the third part is the *Retrieved Knowledge*. Knowledge stored in the rule may be retrieved while reasoning about some action or some description, which is called the *invoking description*.

For example, to select an action that would achieve the result that a given object is made accessible to the actor, the ACT-SELECT rule set is used with an invoking description that indicates which object is to be made accessible to which actor. The Applicability Test is expected to match the invoking description. Then, Retrieved Knowledge is returned only if the World Model Test is successful. Thus the Applicability Test serves to index the rule, whereas the World Model Test tests the situation to see if it is appropriate, and also determines the references to objects in the world model.

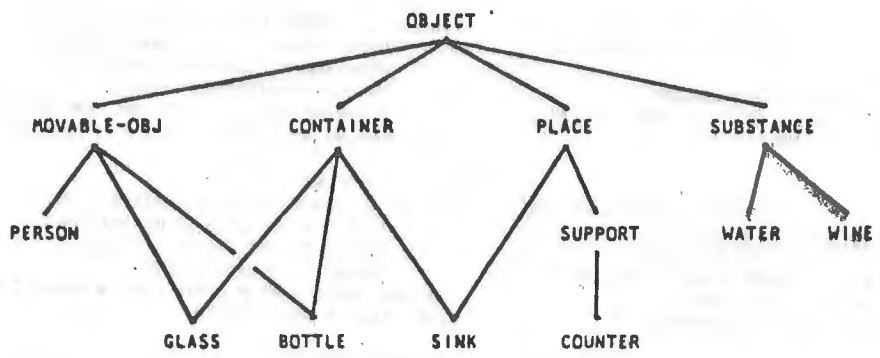
The Applicability Test, serving to index into the rule set, can become expensive to use unless some organizational principles are used to provide *structuring* to avoid irrelevant searching and *inheritance* for economy of expression. We have chosen to organize the rules according to a single partial order relation on the applicability tests, the relation of one description being *more specific than* another. The search for relevant rules starts at the top and prunes any set of rules that are more specific than the one whose applicability test fails (because it is too specific) for the invoking description. The same partial ordering relation is used for providing inheritance, which allows great economy of expression, especially in our description formalism, where a description can be specialized further to an infinite number of other descriptions. Organizing the rules by inheritance also eases the problem of updating rules economically.

There are several ways of defining the relation of *MoreSpecificThan* among descriptions. We provide one such realization, giving the syntax for descriptions (defined in Figure 3) and a brief definition of the relation. A *description form* contains a *description* which mentions variables and an *alist* mapping variables to sets of potential values. A description describes some object, denoted by a variable (or a constant) called the *root* which is a member of a class. For example, (PERSON P (INHAND (CONTAINER C))) has the root variable, P, which is a member of the class PERSON. This object is further specified by a set of relational restrictions, in this example, the person is required to be related to container C by the relation INHAND. When several restrictions are given, the object is to satisfy all of them.

Figure 3: BNF Definition of "DescForm"

```
descform ::= (desc . alist)
   desc ::=  (class root [rspec ...])
      class ::= class-name | class-conjunct
         class-conjunct ::= (class-name ...)
      root ::= variable | constant
      rspec ::= (relation target ...)
         target ::= root | desc
   alist ::= (binding ...) | NIL
      binding ::= (variable value ...)
         value ::= instance | constant
```

Figure 4: Example AKO Hierarchy



There are dimensions over which we have defined our partial order of MoreSpecificThan. A description can be specialized by (i) using a subclass name to denote class membership (based on an explicit AKO (*a kind of*) hierarchy; see Figure 4), e.g. using GLASS rather than CONTAINER, or using the class-conjunct (MOVABLE-OBJ CONTAINER) rather than CONTAINER, (ii) using a constant rather than a variable; (iii) by adding more relational restrictions; (iv) by specializing (recursively) an embedded description, e.g. by using (BOTTLE C (OPEN YES)) in place of (CONTAINER C). In addition to these local factors of the specificity relation, there is the global factor of repetition of variable names. For example, (PERSON P (INHAND (CONTAINER C (OWNEDBY (PERSON P))))) is more specific than (PERSON P (INHAND (CONTAINER C (OWNEDBY (PERSON Q)))))). This definition of the partial ordering is a substantial enrichment of the usual AKO type of structuring and inheritance in vogue.

VI PLANX10

In this section, we describe our implemented planning system, PLANX10 (see [20] for a more detailed description). We have designed an explicit annotated goal tree and plan graph in which the program keeps track of its own problem solving activity. Thus, full state information for the planning process can be saved. The annotated structures also allow us to incorporate distributed planning and incremental planning methods. Furthermore, the same annotation structure is used to make explicit record of assumptions about the presence or absence of objects and structural or functional properties of such objects in the initial situation. The structure of the annotated trace reflects the goal-subgoal structure of the plan, as well as the hierarchical dependencies among the planning decisions and assumptions.

A particular planning algorithm is designed as a collection of planning "meta-actions" that selectively focus attention on parts of the plan graph and goal tree and extend or revise them. User-guidance is one such meta-action. PLANX10 is able to plan with partial descriptions of actions and objects, as well as with phantom objects. Evaluation of a partial constraint description is represented with an alist. That is, the alist associated with a description represents the sets of candidate instances (i.e., variable bindings) which satisfy the constraints of the (possibly partial) description. The selection of a particular candidate instance can be postponed until appropriate constraints have been developed. If information needed for planning is unavailable, PLANX10 is still capable of generating a partial or skeletal plan. If at a later time new relevant information is made available, the plan can be enriched and/or revised.

A. Annotated Goal Tree and Plan Graph

The *Goal Tree* is constructed with three types of Tree Nodes: *GoalNode*, *ActNode*, and *ActionNode*. The nodes are created via the application of an operator to a node; the node operators are described in the next subsection. The node types represent different stages in the plan generation process from goal to subgoal. [See Figure 5.] The Tree Nodes are interconnected in two ways - a *PARENT/PARENTOF relationship* and a two level *OR-AND linkage*. [Figure 6 illustrates an ActionNode.] The OR-AND linkage has the characteristic that if a Node is re-expanded (generating new child Nodes), the goal-subgoal structure is maintained without modifying any previously established links.

Figure 5: Node Operator Mappings

NODE-OPERATOR : Domain ==> Range	
CUSTOMIZE-GOAL	: GoalNode ==> GoalNode
EXPAND-GOAL	: GoalNode ==> GoalNode
SELECT-ACT	: GoalNode ==> ActNode
CUSTOMIZE-ACT	: ActNode ==> ActNode
CREATE-ACTION	: ActNode ==> ActionNode *
VERIFY-PRECONDITIONS	: ActionNode ==> ActionNode
CREATE-SUBGOAL	: ActionNode ==> GoalNode
SIMULATE-ACTION	: ActionNode **

NOTES:
* CREATE-ACTION also creates a PlanNode.
** SIMULATE-ACTION does not create any Nodes.

Figure 6: Example of an ActionNode

```
ActionNode    ACTIONNODE-10
    DescForm      (OPEN X (OF (CONTAINER C))
                    (AGENT (PERSON P)))
                  ((X OPEN-1) (C CABINET1) (P LUKE))
    Outcome
        DescForm      (CONTAINER C (OPEN YES))
                      ((C CABINET1))
    Opportunity
        DescForm      (CONTAINER C (CLOSED YES))
                      ((C CABINET1))
        Type          TRUE
        Support
            State         S1
            Source        NORMALLY-TRUE
    Simulation    S1 ==> S2
    Condition     SUCCESS
    Merit         5
    Parent        ACTIONNODE-7
    OrLink
        AndLink       GOALNODE-11
    CreatedBy     VERIFY-PRECONDITIONS
    LastOperator  SIMULATE-ACTION
```

PLANX10 also generates the *Plan Graph* (composed of PlanNodes). The Plan Graph has cross references with the Goal Tree, and is built upon the ActionNodes extracted from the Goal Tree. A *plan solution* is defined in terms of PlanNodes [see Figure 7]. The knowledge represented by the Plan Graph provides support for the plan critics and activities outside of plan generation (e.g., plan execution, plan recognition, plan revision).

For each action represented in the Goal Tree, there is a PlanNode representing it in the Plan Graph. The PLANOF relation connects a PlanNode to an ActionNode. The other connections involving PlanNodes are defined with reference to this ActionNode:

- A PlanNode ACHIEVES a GoalNode, if the GoalNode represents the *primary goal* of the PlanNode's associated ActionNode.
- A PlanNode X ENABLES a PlanNode Y, if the GoalNode that PlanNode X ACHIEVES represents a precondition of the ActionNode associated with PlanNode Y.
- A PlanNode X SUPPORTS a PlanNode Y, if one of the side effects (i.e., an outcome other than the primary goal) of the ActionNode associated with PlanNode X satisfies a precondition of the ActionNode associated with PlanNode Y.

B. Planning operators

The plan construction actions available in PLANX10 are defined as operators. A node operator is applied to a node to extend and modify the problem solving graph (the combined Goal Tree and Plan Graph). Plan critics are provided to evaluate "global" aspects of the problem solving graph. The critics are applied to subgraphs and perform evaluations and detect various types of conflicts between parallel portions of a plan. The critics are used for such decision tasks as determining the "best" nodes for expansion (i.e., attention focus) and determining the ordering constraints between the actions within a solution plan.

Node operators are shown in Figure 5. These operators connect new nodes to the parent node and add appropriate annotations. CUSTOMIZE-GOAL specializes a GoalNode producing a disjunctive set of GoalNodes. Each GoalNode represents an alternate refinement of the parent goal with respect to knowledge of objects, places, and the current situation. EXPAND-GOAL splits a conjunctive goal into a conjunctive set of simple goals. SELECT-ACT creates a disjuctive set of ActNodes. These represent alternate actions for achieving a given goal. CUSTOMIZE-ACT creates a disjunctive set of ActNodes. Each ActNode is an alternate specialization of the parent ActNode. CREATE-ACTION determines the preconditions and outcomes given a particular action specification (ActNode). The elaborated action specification is associated with an ActionNode and a PlanNode. VERIFY-PRECONDITIONS determines the status (true, false, unknown) of preconditions with respect to a particular situation (i.e. model state). This process may draw conclusions that are based on assumptions about the situation. SIMULATE-ACTION extends the model state history to reflect the outcomes and side-effects of the action simulated. [Figure 8 shows the annotations generated during verification and simulation of an action.] At each step, the user is given the option of directing the planning program. PLANX10 can also be run without user intervention.

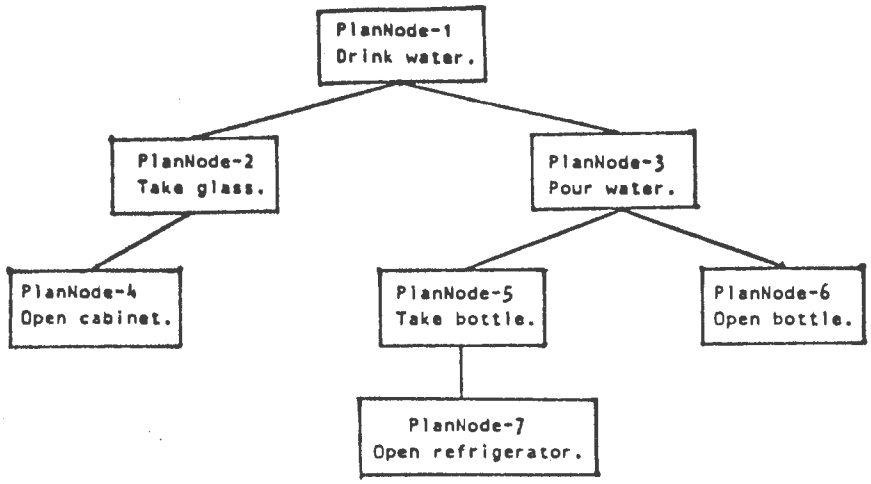
VII Concluding remarks

PLANX10 is programmed in a 23-bit (extended) addressing Lisp, ELISP, using the representation and inference facility AIMDS [21], [22]. PLANX10 has been used in commonsense task domains and in the domain of corporate tax law (e.g., planning a sequence of tax free actions to achieve a desired end state). As our work matures, we expect to report on experimental results in selected large domains.

Acknowledgments: This research was supported by Grant RR-643 to the Rutgers Research Resource on Computers in Biomedicine from the BRP, Division of Research Resources, NIH. We thank Professor C.F. Schmidt and J.L. Goodson for their intellectual contribution to this work, and J.L. Goodson for his help in writing this paper.

Figure 7: Partially Ordered Plan Solution and example PlanNode

Goal : [(PERSON P (DRANK WATER)) . (P LUKE)]



```
PLANNODE     PLANNODE-4
     Enables     PLANNODE-2
     Achieves    GOALNODE-4
                   (MOVE-OBJ O (ACCESSIBLE (PERSON P)))
                   ((P LUKE) (O GLASS1))
     PlanOf      ACTIONNODE-7
                   (OPEN X (OF (CONTAINER C)) (AGENT (PERSON P)))
                   ((X OPEN-1) (C CABINET1) (P LUKE))
```

References

[1] Sridharan, N.S. "Search Strategies for the Task of Organic Chemical Synthesis." In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*. IJCAI, Stanford Research Institute, California, 1973, 95-104.

[2] Stefik, M. *Planning with Constraints*, PhD dissertation, Stanford University, January 1980, (Report STAN-CS-80-784.)

[3] Sacerdoti, E.A. "Problem Solving Tactics." In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*. IJCAI, Tokyo, 1979, 1077-1083.

[4] Sridharan, N.S., Schmidt, C.F., & Goodson, J.L. "The Role of World Knowledge in Planning" In *Proceedings of the AISB-80 Conference*. Society for the study of Artificial Intelligence and Simulation of Behavior, Amsterdam, July, 1980, SRIDHARAN:1-11. (Available as Tech. Rept. RU-CBM-TR-109.)

[5] Schmidt, C.F., Sridharan, N.S., & Goodson, J.L. "Plan Recognition: An Intersection of Artificial Intelligence and Psychology." *Artificial Intelligence, Special Issue on Applications to the Sciences and Medicine*. 11:1 & 2 August (1978) 45-83.

[6] Sridharan, N.S. & Schmidt, C.F., "Knowledge-Directed Inference in BELIEVER" In *Pattern Directed Inference Systems*. Academic Press, 1977, 361-379.

[7] Wilkins, D.E. & Robinson, A.E. "An Interactive Planning System", Technical report 245, SRI International, July 1981.

[8] Vere, S. "Planning in Time: Windows and Durations for Activities and Goals", Technical report, NASA - JPL, November 1981.

[9] Wesson, R. & Hayes-Roth, F. "Dynamic Planning: Searching through Time and Space", Technical report P-6266, Rand Corporation, February 1979.

[10] Sacerdoti, E.D. "Planning in a Hierarchy of Abstraction Spaces." *Artificial Intelligence*. 5:5 (1974) 115-135.

[11] Sacerdoti, E.D. *A Structure for Plans and Behavior*. NY: Elsevier North-Holland, 1977, (Artificial Intelligence Series.)

[12] Thorndyke, P., McArthur, D., & Cammarata, S. "AUTOPILOT: A Distributed Planner for Air Fleet Control", Technical report N-1731-ARPA, Rand Corporation, July 1981.

[13] Srinivas, S. "Error Recovery in Robots through Failure Recovery Analysis." In *Proceedings of the AFIPS National Computer Conference*. AFIPS, Anahiem, CA, 1978, 275-282.

[14] London, P. "Dependency-Based Modelling Mechanism for Problem Solving." In *Proceedings of the AFIPS National Computer Conference*. AFIPS, Anahiem, CA, 1978, 263-274.

[15] Hayes-Roth, B. & Hayes-Roth, F. "Cognitive Processes in Planning", Technical report R-2366-ONR, Rand Corporation, 1978.

[16] Konolige, K. & Nilsson, N.J. "Multiple-Agent Planning Systems." In *Proceedings of the First Annual National Conference on Artificial Intelligence*. AAAI, Stanford, California, August, 1980, 138-142.

[17] Fox, M.S. "Reasoning with Incomplete Knowledge in a Resource-Limited Environment: Integrating Reasoning and Knowledge Acquisition." In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. IJCAI, Vancouver, B.C., August , 1981, 313-318.

[18] Levesque, H.J. "Interaction with Incomplete Knowledge Bases: A Formal Treatment." In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. IJCAI, Vancouver, B.C., August , 1981, 240-245.

[19] Nilsson, N.J. *Principles of Artificial Intelligence*. Tioga Press, 1980.

[20] Bresina, J.L. "An Interactive Planner that Creates a Structured, Annotated Trace of its Operation", Technical report RU-CBM-TR-123, Laboratory for Computer Science Research, Rutgers University, December 1981.

[21] Sridharan, N.S. "AIMDS User Manual, Version 2", Technical report RU-CBM-TR-89, Department of Computer Science, Rutgers University, 1978, (Version 3 forthcoming.)

[22] Sridharan, N.S. "Representational Facilities of AIMDS: A Sampling", Technical report RU-CBM-TM-86, Department of Computer Science, Rutgers University, May 1980.

Planning Within First-Order Dynamic Logic

Henry A. Kautz

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A7

Abstract

This paper develops a version of first-order dynamic logic suitable for stating robot planning problems. The bidirectional planning algorithm developed by S. Rosenschein [12] is extended to handle a subset of this first-order logic, while retaining (provable) correctness. Examples of hierarchical planning, disjunctive and quantified goals, and multiple plan constraints are discussed.

1.0 Introduction

Planning systems based on theorem proving in the situational calculus [5][8] offer a well defined semantics for plans (namely, the semantics of first-order logic), a rich vocabulary for describing world states, and a provably correct planning process. The popular STRIPS-based planners [3] neglect these features in order to concentrate on efficient control of the search through the state-space; state and goal descriptions are limited to sets of literals, which are added or deleted by action operators.

Stanley Rosenschein [12] has formulated the propositional planning problem in dynamic logic, a modal logic developed for program verification [6][10], and has given a complete bidirectional search algorithm ("Bigress") for synthesizing a broad class of plans, including ones involving disjunctive and conjunctive goals and non-deterministic actions. As in the situational calculus, correctness of plans is stated in terms of provability in a formal language; as in STRIPS, plans are found by a structured search through a space of state descriptions.

In this paper, I define a first-order dynamic logic suitable for stating robot planning problems, and extend Bigress to handle a subset of the language. Since state and action descriptions may contain disjunctive and quantified information, calculating the effect of an action on a state (forwards or backwards) is a non-trivial problem. Techniques for query evaluation on first-order data bases are used to determine the instances of the dynamic logic axioms relevant to the calculation of the particular regression or progression.

Perhaps the greatest efficiencies in search can be obtained through the use of plan hierarchies. This is commonly implemented (e.g. ABSTRIPS [13] and NOAH [14]) by temporarily hiding "less important" details from the planning unit; when the details are filled in, the plan may be in error and must be patched up. The complexity of the patching process, unfortunately, weakens our faith in the ultimate correctness of the plan and the overall efficiency of the strategy.

The dynamic logic framework, on the other hand, suggests an exact hierarchy [12]. Solutions to generalized planning problems, containing plan parameters, can be pre-computed. I offer a simple example of provably correct hierarchical planning.

2.0 The Language

A dynamic logic is modal logic: a plan (or program) is a reachability relationship over a set of possible worlds. Where A is a plan, [A]P is true in a world I if P is true in every world reachable from I by A . So [A]P can be read, "after A, P".

Many versions of dynamic logic have been developed (see [2]); generally, assignment is taken as the only basic action, and programs relate worlds with differing interpretations of program variables. We are instead most interested in parameterized actions which affect only the extensions of predicates (e.g. the truth value of on(BLOCK9,TABLE) is changed by the action pickup(BLOCK9)). I designed Transparent Dynamic Logic (TDL) to express such actions.

2.1 Syntax

TDL is an extension of function-free first-order logic. An "action symbol" (e.g. pickup) applied to a sequence of terms is the simplest kind of plan which can appear inside a [ ]. Complex plans are built up by sequencing [A;B] and alternation [P=>A,B] (meaning IF P THEN DO A ELSE DO B). Formally:

Define the sets of symbols:

VAR = variables
CON = constants

PRED(k) = predicate symbols of arity k
(PRED(2) includes "=")
ACT(k) = action symbols of arity k
Let TERMS = VAR U CON

The plans and well formed formulas of TDL are defined recursively:

1. Where a $\in$ ACT(k), $t_1,\ldots,t_k \in$ TERMS, then $a(t_1,\ldots,t_k) \in$ TDL-Plans is an atomic action; the ti's are its parameters.

2. Where P is a non-modal quantifier-free TDL-Wff, and A,B $\in$ TDL-Plans, then null, (A;B), and (P=>A,B) are TDL-Plans.

3. Where p $\in$ PRED(k), $t_1,\ldots,t_k \in$ TERMS, then $p(t_1,\ldots t_k) \in$ TDL-Wffs is an atomic proposition.

4. Where x $\in$ VAR, P,Q $\in$ TDL-Wffs, and A $\in$ TDL-Plans, then ~P, (P v Q), ExP, and [A]P are TDL-Wffs.

A non-modal wff is one not containing a subformula of the form [A]. Define V, &, -> (implies), <-> (equivalent) in terms of E, v, and ~ (not) as usual. A list of terms $t_k,\ldots,x_n$ is often abbreviated $\underline{X}$.

Any plan can be put in "normal form", where B is in normal form if:

i) B can be written A1;A2;...;An with at most one Ai not atomic; and

ii) Such an Ai is of the form (P=>B1,B2) where P is an atomic proposition and B1 and B2 are in normal form.

2.2 Semantics

A structure for TDL includes a a domain of individuals, a set of worlds, and an interpretation for the action symbols. Each world interprets the terms as members of the domain, and the predicate symbols as predicates over the domain. The meaning of a plan is binary relationship over the set of worlds.

These notions are made precise in [7], including semantic conditions for plans to referentially transparent (which allows substitution of terms for variables in modal formulas).

2.3 Axiomatics

The axioms and rules of inference include all those for function-free, first-order logic with equality and loop-free dynamic logic. P and Q stand for any TDL-Wffs, x any variable, B and B' any TDL-Plans, and A for any atomic action. We write P(t/x) for P with t substituted for all free occurrences of x, where [A]P(t/x) = [A(t/x)]P(t/x) (parameter substitution).

first order

A1. All tautologies of propositional calculus
A2. Vx(P ->Q) ->(VxP ->VxQ)
A3. VxP ->P(t/x) , for any term t
A4. P ->VxP , where x is not free in P
A5. Vx(x=x)
A6. $t_1=t_2$ ->(p(...,t_1,....) ->p(...,t_2,...))
any t_1,t2 TERMS, p PRED

modal

A7. [B](P ->Q) ->([B]P ->[B]Q)
A8. [null]P<->P
A9. [B][B']P<->[B;B']P
A10. [P=>B,B']Q<->((P ->[B]Q) & (~P ->[B']Q))

transparency

A12. Vx[A]P ->[A]VxP , x not a parameter of A
A13. Vxy (x=y ->[A] x=y)
A14. Vxy (x≠y ->[A] x≠y)
A15. ~[A] false

rules of inference

R1. From P, P ->Q conclude Q
R2. From P conclude VxP
R3. From P conclude [A]P

Axiom A12, the "Barcan formula" of modal logic, lets us derive a theorem for "pushing" quantifiers through actions:

T1. K[B]P ->[B]KP, where K is a sequence of quantifiers (both E and V), and none of the quantified variables appear in B.

3.0 Finding Plans

Since TDL incorporates first-order logic, it is obviously not decidable. It is well known, however, that the validity problem is decidable for wffs which can be put in prenex form so that all universally-quantified variables precede all existentially-quantified ones [1]. We restrict our attention to a class of planning problems which can be solved using only first-order reasoning about such "V-first" wffs. We classify wffs by their prenex form as "V-first", "E-first", "V-only", or "E-only".

We review and extend the propositional framework from [12].

A planning problem is a triple <VOC, G, R(u($\underline{H}$))> containing:

1. VOC = <CON, PRED, ACT>, the vocabulary of the problem, a finite subset of the symbols of TDL.

2. Domain axioms G, containing:

Gs, a finite set of V-only non-modal (static) axioms.

Gd, a finite set of dynamic axioms, each of the form $(\forall \underline{X})(p \rightarrow [a(\underline{X}')]q)$. $\underline{X}$ and $\underline{X}'$ are lists of variables such that $\underline{X}$ contains $\underline{X}'$, and p and q are non-modal quantifier-free wffs.

The non-modal axioms Gs should be strong enough to derive all non-modal theorems derivable from G.

3. Plan constraints $R(u(\underline{H}))$, a finite set of wffs of the form $(\forall \underline{X})(r \rightarrow [u(\underline{H})]s)$, where r is non-modal and $\forall$-only, s is non-modal and $\exists$-only, and $\underline{H}$ is a distinguished set of variables called the "problem parameters". "u" is a special action symbol which does not appear in VOC.

A solution is a plan B such that substituting B for $u(\underline{H})$ in each plan constraint creates a theorem derivable from the axioms ($G \vdash r \rightarrow [B]s$ for each plan constraint), and the only free variables in B are problem parameters. $\underline{H}$ is the "input" to the plan.

For example, if $R(u(\underline{H}))$ is

$$\{\ \forall h(block(h) \rightarrow [u(h)]inbox(h)),\ \forall hx((\neg inbox(x) \wedge x \neq h \rightarrow [u(h)]\neg inbox(x))\ \}$$

the constraint is to find a plan to put any block h into the box, without putting anything else in the box. The only input to the plan will be the particular block to be moved.

The basic strategy of Rosenschein's planning algorithm is as follows. We define the strongest provable postcondition of a non-modal wff r and an action A as a non-modal formula r/A such that $G \vdash r \rightarrow [A]r/A$ (r/A is a postcondition) and $G \vdash r/A \rightarrow q$ whenever $G \vdash r \rightarrow [A]q$. Similarly, the weakest provable precondition of a wff s and action A is a non-modal formula A\s such that $G \vdash A\backslash s \rightarrow [A]s$ and $G \vdash q \rightarrow A\backslash s$ whenever $G \vdash q \rightarrow [A]s$.

Since $\vdash \forall X(r \rightarrow [B]s)$ iff $\vdash r \rightarrow [B]s$, we can drop the quantifiers which quantify over the plan constraints. The algorithm first checks whether each initial state description (antecedent in the plan constraints) implies the corresponding goal description (consequence in the plan constraints); if so, the search halts. Otherwise, it non-deterministically chooses to either insert a conditional branch in the plan under construction, or to select a simple action A, and either progress each initial state description through it (calculating r/A), or regresses each goal state description through it (calculating A\s). The process is then applied recursively.

In the following algorithm, braces < > are used to construct lists of syntactic variables or formulas. Where pre_i and $post_i$ for $1 \leq i \leq n$ are syntactic variables ranging over wffs, we write:

PRE for $\langle pre_1, \ldots, pre_n \rangle$
POST for $\langle post_1, \ldots, post_n \rangle$
PRE/A for $\langle pre_1/A, \ldots, pre_n/A \rangle$
A\POST for $\langle A\backslash post_1, \ldots, A\backslash post_n \rangle$
PRE & T for $\langle pre_1 \& T, \ldots, pre_n \& T \rangle$

A solution to the planning problem <VOC, G, R(u(H))> where $R(u(\underline{H}))$ is

$$\{\ \forall x1\ (r_1 \rightarrow [u(\underline{H})]\ s_1),\ \ldots\ \forall xn\ (r_n \rightarrow [u(\underline{H})]\ s_n)\ \}$$

is found by calling
Bigress($\langle r_1,\ldots,r_n \rangle$, $\langle r_1,\ldots,s_n \rangle$, null, null).

Bigression Algorithm *

```
Bigress(PRE, POST, leader, trailer):

  IF G|-pre_i ->post_i for 1≤i≤n THEN
     RETURN(leader;trailer).

  CHOOSE:

    CHOOSE <A, PRE/A> FROM LiveForward(PRE):
         RETURN(Bigress(PRE/A, POST, leader;A,
                                     trailer)).

    CHOOSE <A, A\POST> FROM LiveBackward(POST):
         RETURN( Bigress(PRE, A\POST, leader,
                                   A;trailer)).

    CHOOSE T FROM NonTriv(PRE):
         RETURN( leader; C; trailer ) where
  C = (P==>Bigress(PRE & T,  POST, null, null),
         Bigress(PRE & ~T, POST, null, null)).
end.

LiveForward(PRE):
   RETURN { <a(t_1,...,t_k), PRE/a(t_1,...,t_k)> |
            a ∈ ACT(k), ti ∈ CON U H, and
            for SOME i, 1≤i≤n,
            not G|- pre_i ->pre_i/a(t_1,...,t_k) }

LiveBackward(POST):
   RETURN { <a(t_1,...,t_k), a(t_1,...,t_k)\POST> |
            a ∈ ACT(k), ti ∈ CON U H, and
            for SOME i, 1≤i≤n,
            not G|- a(t_1,...,t_k)\post_i ->post_i }

NonTriv(PRE):
   RETURN { p(t_1,...,t_k) | p ∈ PRED(k),
         ti ∈ CON U H, for SOME i, 1≤i≤n,
         not G|- pre_i ->T
         not G|- pre_i ->~T }
```

By making the non-modal axioms strong enough to generate all non-modal theorems, the stopping test $G \vdash r \rightarrow s$ holds if and only if $Gs \rightarrow (r \rightarrow s)$ is a valid formula of first-order logic with equality; the latter formula is $\forall$-first, and so (as in the propositional case) the test is decidable. The tests in LiveForward,

---

* Adapted from the single constraint, propositional Bigress algorithm in [12].

LiveBackward, and NonTriv, also only applied to ∀-first formulas, serve to prune loops in the search space. For instance, if G|-r ->r/A for every plan constraint, no solution need begin with A, because A;B would satisfy the plan constraints only if the shorter plan B would as well.

4.0 Examples

Before discussing the progression and regression operators in detail, we offer a number of example problems that can be solved by Bigress, but would prove difficult for STRIPS and its more sophisticated descendents.

Several boxes can hold various colored stones. The "dump" action transfers all the stones that are in one box to another. The static axioms describe basic facts which are true in every state, and the dynamic axioms describe (perhaps only partially) "dump".

```
VOC: CON:  B1, ..., B3, S1, ..., S10
     PRED: box, in, color
     ACT:  dump

Gs:  box(B1), box(B2), box(B3)
     B1≠B2, B1≠B2, ..., S9≠S10

Gd:  ∀xyz((in(x,y) & box(z))
        ->[dump(y,z)] in(x,z))

     ∀wxyz((in(x,w) & w≠y)
        ->[dump(y,z)] in(x,w))

     ∀wxyz(~color(w,x)
        ->[dump(y,z)] ~color(w,x))
```

The last two axioms are frame axioms: axioms which describe conditions which are invariant under an action [8]. STRIPS avoids explicit frame axioms by maintaining a single model of the world, with the understanding that any proposition not explicitly deleted by an action operator carries through from the previous state. But an action like dump, which affects an arbitrarily large number of objects, can (as Waldinger [16] notes) nullify any efficiency advantage gained by such a strategy: the system still has to check and possibly add or delete a great many propositions.

Problem 1: demonstrates quantified, conjunctive, and disjunctive state descriptions and goals. The plan constraint:

```
∀x ((∀y(color(y,BLUE) ->
              (in(y,B1) v in(y,B2)))
     & in(S2,B3)
      ->[u]((color(x,BLUE) ->in(x,B2)) &
            ∃z(in(z,B1)))
```

which can be read, "given that all blue things are in either B1 or B2, and S2 is in B3, get all blue things in B2, and something in B1" is solved by:

```
dump(B1,B2);dump(B3,B1)
```

Disjunctions appear explicitly in the constraints antecedent, and implicitly in the implication in the consequence. "All blue things are in B2" cannot be expressed by a set of literals, as we have not invoked any "closed world assumption" about the set of blue things, nor need we have specific constants to name them.

Problem 2 is a variant of the "register exchange" problem [16], and illustrates the dispatch of Bigress's regressive search.

```
(in(S1,B1) & in(S2,B2))
   ->[u](in(S1,B2) & in(S2,B1))
```

The solution

```
dump(B1,B3);dump(B2,B1);dump(B3,B2)
```

is found with no backtracking. LiveBackward(in(S2,B1) & in(S1,B2)) contains only dump(S3,S2) and dump(S3,S1); other choices are pruned, since the goal regresses through them to false. Either possible choice leads directly to the solution.

Straightforward linear planners (e.g. original STRIPS, which tries to achieve each conjunct of a conjunctive goal in sequence) fail on this problem, since a first step of dump(B1,B2) renders the solution impossible. A non-linear planner (e.g. NOAH [14], which creates independent subplans to achieve each conjunct) finds that its subplans can't be combined, and must replan. RSTRIPS [16] is famed for its similar regressive solution to this problem; but in more complicated cases, Bigress's stronger handling of disjunctive goals (as regressions are generally disjunctions) could be more efficient.

Example 3: involving a non-deterministic action, is a version of the ubiquitous "3-socks" problem: how to find a pair of matching socks in a dark room? Say S1,...,S10 are black or white socks, initially in B1; "take" picks one out at random, and "put" puts it in B2; the goal is to have a matching pair in B2.

```
Gs:
 ∀xy(hold(x) ->~in(x,y))
 ∀x(sock(x) ->
    (color(x,BLACK) v color(x,WHITE)))
 sock(S1), ..., sock(S10)

Gd:
 ∀x(in(x,B1)
     ->[take](hold(S1) v ... v hold(S10))

 ∀x(in(x,B1) ->[take](hold(x) v in(x,B1))

 ∀x(in(x,B2) ->[take]in(x,B2))
```

```
∀x(hold(x) ->[put]in(x,B2))

∀xy(in(x,y) ->[put]in(x,y))

R(u):
 (in(S1,B1) v ... v in(S10,B1)) ->[u]
   Exy(in(x,B2) & in(y,B2) & x≠y &
       ((color(x,BLACK) & color(y,BLACK)) v
        (color(x,WHITE) & color(y,WHITE))))
```

Bigress grinds out the solution: dip into the box just three times.

```
take;put;take;put;take;put
```

Of the other systems discussed here, only NOAH can be programmed for "pick up a random sock"; but its weaker handling of disjunctive goals would lead it to create separate subplans to try to achieve either "two black socks are in B2" or "two white socks are in B2" -- neither of which can be realized under the given axiomatization.

5.0 Progression and Regression

The next task is to define the progression / and regression \ functions. We concentrate here on the former; regression is handled similarly (see [7]). The following algorithm is proven correct in detail in [7], but with a hitch: in certain cases it may not terminate; pathological examples can be constructed where the strongest provable postcondition (weakest provable precondition) cannot be represented by a finite length non-modal formula (see section 7.0).

The progression of a state description r through an atomic action a(Z) using domain constraints G is calculated in four steps. The basic idea is find all the substitution instances of the dynamic axioms for a(Z) whose antecedents are implied by r. A simple example is worked out in parallel to illustrate the details.

Example: Calculate the first progression found by Bigress in a forward search for the solution to problem (1) above:

```
(∀x(color(x,BLUE) ->(in(x,B1) v in(x,B2)) &
 in(S1,B3)) / dump(B1,B2)
```

Step 1: Form G(a(Z)), the set of instances of all the dynamic axioms for a, including A13 and A14 (the equality frame axioms) with parameter list Z.

Ex: G(dump(B1,B2)) =

```
{ ∀x((in(x,B1) & box(B2)) ->
            [dump(B1,B2)] in(x,B2)),

  ∀wx((in(x,w) & w≠B2) ->
            [dump(B1,B2)] in(x,w)),

  ∀wx(¬color(w,x) ->
            [dump(B1,B2)]¬color(w,x)),

  ∀wx (w=x ->[dump(B1,B2)] w=x),

  ∀wx (w≠x ->[dump(B1,B2)] w≠x) }
```

These formulas are all derivable from G by A3.

Step 2: Replace any free variables in r and G(a(Z)) by new constants; call Gs and the transformed versions of r the "base formulas".

Define a "query" as an E-quantified non-modal formula in disjunctive form. A "variant" of a formula is created by substituting various terms for its E-quantified variables. An "answer" to a query is a disjunction of of variants of some of the disjuncts of the query, such that the answer is derivable from the base formulas.

Form a query by taking the disjunction of the antecedents of each formula in G(a(Z)), placing E-quantifiers on the variables.

Ex: The base formulas are just Gs and r above. The query consists of five disjuncts:

```
Ex (in(x,B1) & box(B2)) v
Ewx (in(x,w) & w≠B1) v
Ewx(¬color(w,x)) v
Ewx (x=w) v
Ewx (x≠w)
```

Step 3: Generate and conjoin (at least) all significantly different answers to the query. Resolution with answer-extraction provides a complete strategy for answer generation based on theorem proving [4]. The equality axioms (A5-A6) can be simulated by by special rules of inference, such as paramodulation [11]. Complete heuristics such as subsumption [2] can be used to avoid generating weaker and weaker versions of the same answer (e.g., don't generate p(C) or p(x) v q as answers if p(x) has already been generated). Disregard answers derivable from Gs alone which are variants of the frame-axiom antecedents.

Ex: Answers to the query include:

```
[1] ∀x((in(x,B1) & box(B2) v
        (in(x,B2) & B2≠B1) v ¬color(x,BLUE))
    &
[2] (in(S1,B3) & B3≠B1)
```

We disregard answers such as "S1=S1" which don't depend on r; their inclusion would merely complicate the calculation. The notion is to try to avoid including in the final postcondition facts which are already in the domain constraints.

Step 4: Replace each variant p' in the conjunction of answers by the corresponding variant q'; that is, where θ is a substitution

and $\forall \underline{X}(p \rightarrow [a(\underline{Z})]q)$ a member of $G(a(\underline{Z}))$ such that $p' = pQ$, let $q' = qQ$.

Free variables in the resulting formula take on universal quantifiers. Finally, the new constants introduced in step (2) are replaced by the original free variables.

Ex: The transformed formula yields:

$$\forall x(in(x,B2) \vee in(x,B2) \vee \neg color(x,BLUE)) \;\&\; in(S1,B3)$$

as the final postcondition; or simplifying,

$$\forall x(color(x,BLUE) \rightarrow in(x,B2)) \;\&\; in(S1,B3)$$

That this formula holds after dump(B1,B2) follows from the fact the given state description implies the conjunction of answers (by the deduction principle of first-order logic). Propositional modal reasoning lets us combine the antecedents and consequences of non-quantified instances of the dynamic axioms for $a(\underline{Z})$; axiom A12 allows the $\forall$-quantifier on x to be transfered across [dump(B1,B2)].

Given that r is $\forall$-only and s E-only, r/A and A\s are $\forall$-only and E-only respectively as well. Thus the quantifier rules imposed on the parameters of Bigress to ensure decidability of the provability test G|- are maintained.

6.0 Hierarchical Planning

The search performed by a planner tends to grow exponentially as the length of the solution increases; the computational problem is exacerbated by the very flexibility a purely logical system allows in representing state descriptions and actions. The dynamic logic framework suggests an elegant and natural framework for the hierarchical decomposition of a planning problem into smaller subtasks.

A hierarchical planning problem is a tree of single-level problems [12]. Higher level domain dynamic axioms are simply taken as lower level plan constraints. Specifically, for each atomic action $a(\underline{Z})$ which appears in a solution to a node $\langle VOC_i, G_i, R_i(u(\underline{H}))\rangle$, there is a child problem $(VOC_{i+1}, G_{i+1}, Gi(a(\underline{Z}))\rangle$, where ACT_{i+1} includes actions more primitive than those in ACT_i. Because the lower-level plan which implements the higher-level action satisfies the plan constraints derived from the higher-level frame axioms, it is free from unexpected side-effects; and so the overall plan is constructed by simply piecing together the lowest-level solutions.

The i+1 level planner can either solve for the specific instances of the actions used by the higher-level plan, or compute a more general solution which takes the parameters of the high-level action as input. The relevant theorem is:

T2. Suppose $\langle VOC, G, G'(a(\underline{Z}))\rangle$ has solution B. Then for any substitution Q of terms for variables, $\langle VOC, G, G'(a(ZQ))\rangle$ has solution BQ.

For example, take problem (1) above as a top level planning problem, <VOC1, G1, R1(u)>. No mention has been made of how the dump action actually transfers items from box to box. Level 2 introduces a robot that can grab all the objects in a box, move from place to place, and drop the items it's carrying.

VOC2: CON2: (as above)
PRED2: box, in, robat, hold, color
ACT2: goto, grab, drop

Gs2: $\forall$ loc obj(in(loc,obj) ->box(obj))

Gs contains

$\forall$loc(box(loc) ->[goto(loc)]robat(loc))

$\forall$ loc obj((in(loc,obj) & robat(obj)) ->[grab]hold(loc))

$\forall$ loc obj((hold(loc) v robat(obj)) ->[drop]in(loc,obj))

plus appropriate frame axioms for the three actions.

Rather than treat the constraint sets G1(dump(B1,B2)) and G1(dump(B3,B1)) as specifying unrelated planning problems, the planner can solve the more general problem

<VOC2, G2, G1(dump(y,z))>

whose three plan constraints are exactly the main and frame axioms in G1 for dump, and y and z are plan parameters.

From the solution to this problem,

goto(y);grab;goto(z);drop

T2 above lets us derive the appropriate plans to implement dump(B1,B2) and dump(B3,B1). The overall solution is in the level 2 vocabulary, and satisfies the level 1 plan constraints:

```
G2|-∀x((∀y(color(y,BLUE) ->
              (in(y,B1) v in(y,B2)))
      & in(S1,B3)) ->
      [goto(B1);grab;goto(B2);drop;
       goto(B3);grab;goto(B1);drop]
         ((color(x,BLUE) ->in(x,B2)) &
          Ez(in(z,B1))))
```

The search involved in finding a high-level 2-step plan and a low-level 4-step plan is orders of magnitude less than that needed to directly synthesize the overall solution.

7.0 A Problematical Case

The quantifier-order rules are not quite sufficient to guarantee termination of the answer-generation process in the progression or regression functions. Although we admit only a finite number of constant terms (and no functions), arbitrarily long significant answers containing (V-quantified) variables can arise in the presence of "recursive" domain axioms. For example, let G contain only:

```
∀xy  (on(x,y) ->above(x,y))
∀xyz ((on(x,y) & above(y,z)) ->above(x,z))
∀xy  (¬on(x,y) ->[a] ¬on(x,y))
```

Note that with this axiomatization, it is not provable from G that ¬above(K1,K2) ->[a]¬above(K1,K2). In attempting to calculate ¬above(K1,K2)/a, the answer generation process is bound to produce an infinite series of answers corresponding to the fact that the strongest provable postcondition of ¬above(K1,K2) can only be written as the infinite conjunction

```
¬on(K1,K2)
& ∀x (¬on(K1,x) ∨ ¬on(x,K2))
& ∀xy (¬on(K1,x) ∨ ¬on(x,y) ∨ ¬on(y,K2))
& . . .
```

The damage can be repaired in the present case by augmenting G with a frame axiom for ¬above, so that ¬above(K1,K2) is a postcondition which implies each conjunct of the above expression. The exact constraints on G to prevent such problems in the general case remain a topic of current work. It is important to note, however, that even if the answer generation process used by / and \ is incomplete, any solutions found by Bigraas will still be provably correct, although, of course, some answers may be lost.

8.0 Conclusions and Extensions

The system described has not yet been implemented here at Toronto. The world doesn't need another program that solves block stacking problems; an incrementally extensible theory of planning is more interesting. This paper describes an early step in this process. Its major contribution is to provide a first-order language for planning and a provably correct method for handling progression and regression in a complex world.

Efficiency considerations would probably preclude a direct implementation of all the frame axioms, and demand a special treatment of the equality predicate.

[7] suggests how Bigraas could use "formal objects" [15] to reduce search time.

An important extension is the incorporation of loops -- necessary, for example, if the dump action is to be implemented by a low level plan involving a robot moving objects one at a time. Much work needs to be done on the hierarchical planning aspects, viz: what characterizes a "good" hierarchical decomposition of a problem domain? How much detail can the hierarchy suppress, while avoiding bad interactions between subplans? How exactly can a dynamic hierarchy (as in [14]) be formalized in dynamic logic?

References

[1] Ackermann, Wilhelm (1954) Solvable Cases of the Decision Problem.

[2] Chang, C.L. and Lee, R.C.T. (1973) Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York.

[3] Fikes, R.E. and Nilsson, N.J. (1971) STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3/4), 189-208.

[4] Green, C. (1969) Theorem-proving by resolution as a basis for question-answering systems. Machine Intelligence 4, Edinburgh University Press, Edinburgh.

[5] Green, C. (1969) Application of theorem proving to problem solving. International Joint Conference on Artificial Intelligence, (D. Walker and L. Norton, Eds.) Washington, D.C.

[6] Harel, David (1979) First Order Dynamic Logic, Springer-Verlang.

[7] Kautz, Henry (1981) A Dynamic Logic for Planning, (M.S. thesis), University of Toronto.

[8] McCarthy, J. and Hayes, P. (1969) Some philosophical questions from the standpoint of artificial intelligence. Machine Intelligence 4, Edinburgh University Press, Edinburgh.

[10] Pratt, Vaughan R. (1976) Semantical Considerations on the Floyd-Hoare Logic. Proceedings of the 17th IEEE Symposium on the Foundations of Computer Science.

[11] Robinson, G.A. and Wos, L. (1969) Paramodulation and Theorem Proving in First Order Theories with Equality. Machine Intelligence 4, (B. Meltzer and D. Michie, Eds.) American Elsevier, New York.

[12] Rosenschein, Stanley J. (1981) Plan synthesis: a logical approach. Proceedings of the 8th International Joint Conference on Artifical Intelligence, University of British Columbia, Vancouver, BC.

[13] Sacerdoti, E. (1974) Planning in a hierarchy of abstraction spaces. Artifical Intelligence, 5(2), 115-135.

[14] Sacerdoti, E. (1974) A Structure for Plans and Behavior, American Elsevier, New York.

[15] Sussman, G.J. (1975) A Computer Model of Skill Aquisition, American Elsevier, New York.

[16] Waldinger, R.J. (1977) Achieving several goals simultaneously. Machine Intelligence 8: Machine Representations of Knowlege, Ellis Horwood, Chichester.

INTERPRETATION BASED INTERACTION BETWEEN LEVELS OF DETAIL

Roger A. Browse

Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada V6T 1W5

ABSTRACT

This paper discusses the operations of a computer vision system used to interpret line drawings of body forms. Mechanisms are described which permit interactions among information obtained at different levels of resolution. These mechanisms operate in a context of availability of information which is similar to that of the human vision system.

1. INTRODUCTION

Interaction between different levels of detail (resolution or spatial frequency) play an important role in computational vision. For the most part, the operations involving these different levels do not relate to the scene domain knowledge being used in interpretation (Hanson and Riseman, 1974; Tanimoto, 1976).

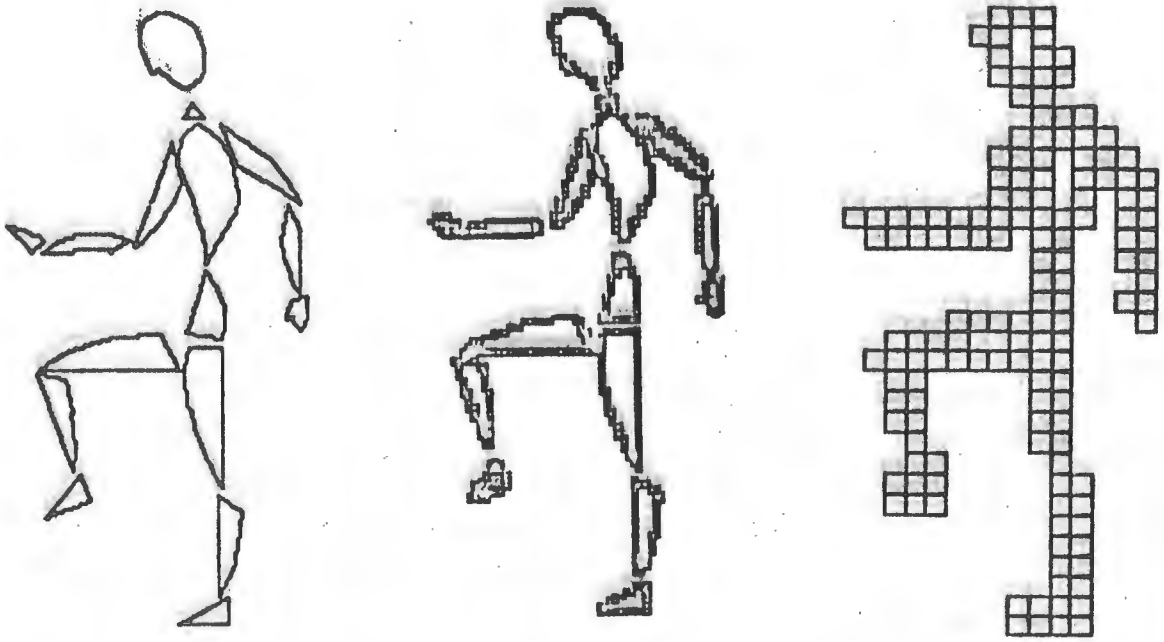
On the other hand, Kelly (1971) and Shirai (1973) have shown that each level of detail may be separately interpreted and that the result of one level may aid the interpretation of the other. Rosenthal and Bajcsy (1978; Bajcsy and Rosenthal, 1980) describe how search for specific objects in an image can be assisted by being able to predict the level of detail at which an object can be located by exploiting the containment relation between scene domain objects.

This paper describes the operations of a computational vision system which permits interpretation of images through cooperative interaction among information obtained from more than one level of detail. It has been previously shown that this approach provides compatibility with aspects of human vision (Browse, 1981). This paper outlines the system's ability to reduce interpretation possibilities by providing scene to image domain mappings at two levels of detail.

2. SYSTEM DESCRIPTION

The vehicle for this research is a system to interpret line drawings of body forms. The examples being used are taken from Eshkol and Wachmann (1958) (see figure 1a). The design of the system adheres to the idea that domain knowledge should be declarative, and separate from the interpretation methods (see Browse, 1980). As a means of testing the adequacy of the scene domain knowledge, it has been translated into PROLOG, and used to "prove" body forms in a data base of image assertions.

The most suitable scene domain based interpretation schemes are found in the schemata based systems of Mackworth and Havens (1981; Havens and Mackworth, 1980). Provision for a recursive cuing mechanism is an integral part of schemata based systems. Thus the body form



1a. 1b. 1c.

figure 1. display of body-form at different levels of detail

knowledge has been translated into a representation suitable for such operations which is rich in component and specialization hierarchies.

2A. Image Operations

A line drawing is input as in figure 1a, with the aid of a light pen on a graphics terminal. Subsequently an image pyramid is developed, with each stage in the pyramid representing a different level of detail (see figures 1a to 1c). The pyramid represents the entire ambient array, but at any instant during interpretation, fine detail information is available in a small area of the image, with a surrounding area of coarser information in a way analogous to the availability of information to the human eye (see figure 1d).

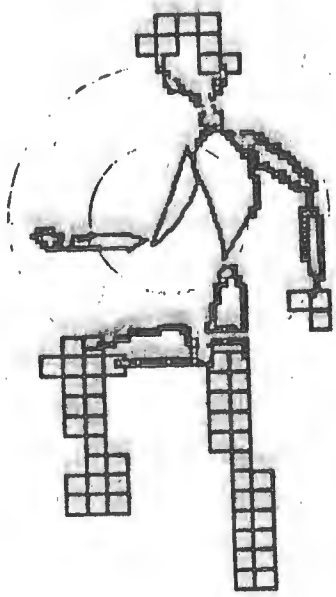


figure 1d. Information available within fixation

Primitive image elements are available at different levels of detail. Blobs with determinable characteristics are available from the coarse levels of detail, while lines and line connections are available from the fine detail level.

2B. Interpretation

Image primitives from any level can act as cues to interpretation possibilities. For example, a blob with certain properties may suggest any of a number of body parts (such as hand, foot, or head), while a line connection of a certain type may suggest particular views of a hand. This cuing function is always one-to-many and thus the organization of primitives is critical to understanding the image. In general, fine detail image elements (lines) cue scene elements at the fringe level of the component and specialization hierarchies, whereas coarse detail image elements (blobs) cue elements higher (towards the root node) in the hierarchies.

If only coarse level image elements were being extracted, the system could still effect interpretation, and similarily for fine detail image elements, though the resulting image description would be richer if the fine detailed information were used.

There are a number of ways that multiple levels of interpretation can operate at the same time, cooperating and sharing their ongoing results through a common scene domain representation. By keeping the scene domain knowledge specification separate from the interpretation process, it is possible to investigate a number of such possibilities. The following section describes a data-driven method currently being considered.

3. AN EXAMPLE OF MULTI-LEVEL INTERPRETATION

The interpretation process draws a sharp distinction between the detection of an image primitive (feature), and the analysis of its relation to other elements (feature integration). Within the information available at a time, the system attempts to delay the integration step. This provides a point of interaction between levels of interpretation and it is also consistent with the fact that in human vision, features may be detected rapidly, accurately, and in parallel over a stimulus, whereas the integration of these features requires the sequential application of a attentional mechanism to the location being integrated (Treisman and Gelade, 1980).

First of all, image primitives are extracted from the coarse level of detail. Each detected primitive maintains a set of possible models of which it may be a part. Before considering the relations among these blobs, the fine detail area is processed for its image primitives. Each fine detail primitive also has a set of possible models (see figure 2).

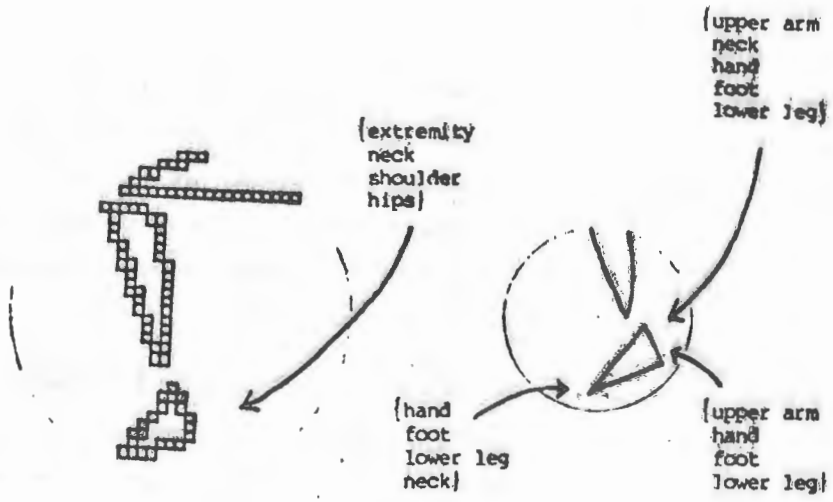


Figure 2. Model possibilities at two levels of detail

The image hierarchy structures may provide the knowledge that some set of fine level primitives correspond with some coarse level primitive. In this case, two types of preliminary interpretation filtering may take place.

1. Since the set of fine level primitives must belong to the same model, inconsistent interpretation possibilities may be eliminated (as shown in figure 3).

2. Interpretation at the coarse level must be the same, or a generalization of, the interpretation at the fine level. Therefore more inconsistent possibilities can be eliminated (as shown in figure 4).

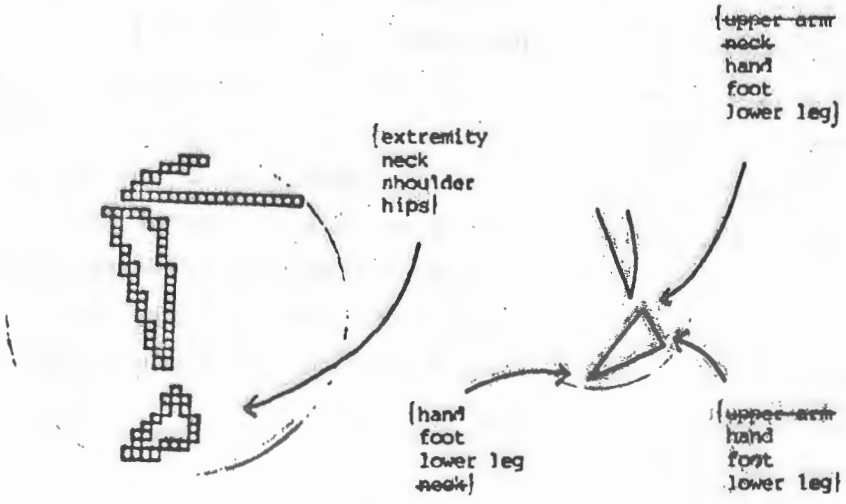


Figure 3. Model possibilities after within-level filtering

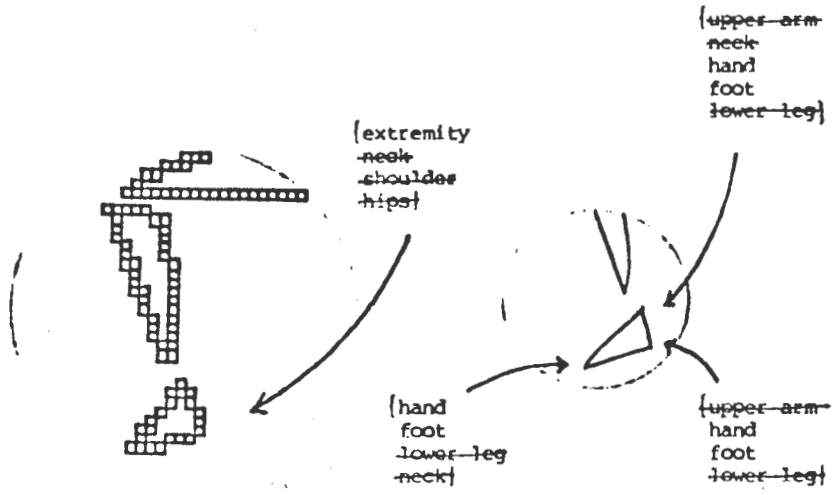


Figure 4. Model possibilities after across-level filtering

Each blob which lies within the fixation area is processed in this way, resulting in a significant reduction in the number of possible interpretations for the primitive image elements. For the example fixation shown in figure 1d., the application of these two filtering operations eliminates 60% of the possibilities.

The next step involves the activation of upper level models in a way similar to that of Havens and Mackworth (1980). This model activation considers the relations among primitives at each level, and establishes the existence of more complex scene concepts. In the periphery, blobs may be interpreted as such general concepts as "extremity" or "upper-limb". In the foveal area, more detailed interpretations are found, involving specific body parts with restricted ranges of three dimensional orientation.

Through the use of the specialization hierarchy within the body model knowledge, the more detailed interpretation at the fovea may propogate outwards to force instantiation of the peripheral objects.

4. CONCLUSIONS

This paper has proposed the approach of allowing information from different levels of detail to interact through a common goal of interpretation, utilizing a common scene domain knowledge. An example data-driven interpretation method is provided which demonstrates a computational advantage to the approach, and which also shows consistency with some established aspects of human vision. Other interpretation schemes, using the same scene domain knowledge are planned. These will consider the possibility of inadequate segmentation at the coarse level of detail, and will consider the development of intelligent decisions regarding the subsequent placement of the retina on the ambient array representation.

5. REFERENCES

Bajcsy, R. and Rosenthal, D.
1980. Visual and Conceptual Focus of Attention, in Structured Computer Vision, S. Tanimoto and A. Klinger (eds), Academic Press, N.Y.

Browse, R.A.
1980. Mediation Between Central and Peripheral Processes: Useful Knowledge Structures, CSCSI-3, pp.166-171.

Browse, R.
1981. Relations between Schemata-Based Computational Vision and Aspects of Visual Attention, PROC Third Conf. Cognitive Science Society, Berkeley, Ca., pp.187-190.

Eshkol, N. and Wachmann, A.
1958. A Movement Notation, Weidenfield and Nicolson, London.

Hanson, A. and Riseman, E.
1974. Preprocessing Cones: A Computational Preprocessor. Tech. Report 74C-1 Computer and Information Sciences, University of Mass.

Havens, W.S.
1978. A Procedural Model of Recognition for Machine Perception, Ph.D. Thesis, Dpt. Computer Science, University of British Columbia (TR-78-3).

Havens, W.S. and Mackworth, A.K.
1980. Schemata-based Understanding of Hand-Drawn Sketch Maps, CSCSI-3, pp. 172-178.

Kelly, M.D.
1971. Edge Detection by Computer Using Planning, Machine Intelligence, Vol 6, pp. 379-410.

Mackworth, A.K. and Havens, W.S.
1981. Structured Domain Knowledge for Visual Perception, IJCAI-81, Vancouver, Canada, pp. 625-627

Rosenthal, D. and Bajcsy, R.
1978. Conceptual and Visual Focussing in the Recognition Process as Induced by Queries, IJCPR-4, pp. 417-420.

Shirai, Y.
1973. A Context Sensitive Line Finder for Recognition of Polyhedra, Artificial Intelligence, 4, pp. 95-199.

Tanimoto, S. L.
1976. Pictorial Feature Distortion in a Pyramid, Computer Graphics and Image Processing, Vol 5, Sept 1976.

Treisman, A. and Gelade, G.
1980. A Feature Integration Theory of Attention, Cognitive Psychology, 12, pp. 97-136.

A Schemata-Based System for Utilizing Cooperating Knowledge Sources in Computer Vision

Jay Glicksman

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5

Abstract

Computational vision is difficult because of ambiguous, incomplete, and inconsistent data. An approach to these problems is a system which supports multiple interpretations, uses a world model, and incorporates several types of input information. The data structures for a schemata-based system having these properties are described with some examples of how they can be employed.

1. Introduction

Perception of real-world phenomena must start with some iconic representation of the world. Unfortunately, elements of the image relate to the elements in the scene only through some unknown mapping which is confounded by the imaging process. Factors such as lighting, viewpoint, exposure, the focal length of the lens, and surface reflectance complicate the understanding of the image. As individuals, the image elements (pixels, edge elements, etc.) convey no meaning (unlike, say, words in text): the meaning can emerge only during the perceptual process.

This leads to three undesirable situations which must be remedied if effective interpretation is to take place. Ambiguous situations arise when data can be interpreted in more than one way. Incompleteness occurs when one does not have enough data to conclusively support an interpretation. Finally, inconsistency is the result of uncovering evidence both for and against an interpretation.

One way to resolve ambiguities is to add sources of information to the original image. Examples include range data (Nitzan et al., 77), elevation data (Horn and Bachman, 78), and the work on map-guided interpretation at SRI (Tenenbaum et al., 78). Adding information sources has the advantage of augmenting the system's knowledge, but can lead to more inconsistencies.

When interpretations can be matched to a world model, incompleteness can often be accommodated. Also, a world model could potentially resolve ambiguities and inconsistencies in terms of what is important in the model. In the aforementioned SRI system, the map can be considered a primitive world model. More elaborate examples include models of geographic regions (Bajcsy and Tavakoli, 73; Havens and Mackworth, 80).

Handling inconsistent knowledge generally requires a flexible system that permits multiple interpretations and non-monotonic reasoning. Such a system will form the best interpretation based on the current context. Ambiguities could also be

resolved by giving precedence to the most contextually "reasonable" interpretation.

The Hearsay II speech understanding system (Erman and Lesser, 75; Lesser and Erman, 77) exhibits the type of flexibility needed to handle inconsistent information. Of particular note is the use of knowledge sources which are specialists in analyzing some part of the domain. They provide the proper context to deal with incomplete data. Knowledge sources are written as productions in a production system.

Another current mechanism of similar modularity is based on units of schemata. Mapsee2 (Havens and Mackworth, 80; Mackworth and Havens, 81) is an example of a schemata-based system using a geographic world model. It interprets line drawings as sketch maps. The following section describes an extension of Mapsee2 which uses several types of input information. In particular, the extension supports the use of a sketch map and a grey-scale image, each of which can aid in the interpretation of the other.

2. A Representation for Schemata

The data structures for the system being developed (called Maids) are an extension of Maya (Havens, 78), and use both Maya functions and the base language, UBC Multilisp (Koomen, 80). In addition, Maids incorporates a number of ideas from FRL (Roberts and Goldstein, 77a and 77b).

Schemata are the basic representation units in Maids. They are composed of a list of attribute-name and attribute-value pairs. There are four distinguished categories of attributes: LINKs, VALUEs, PROCEDUREs, and CONFIDENCE.

The knowledge base is a collection of intertwined schemata hierarchies, with relationships indicated by LINKs. Any number of relations can be defined, such as LINKs to form a specialization (or subset) hierarchy, a decomposition (or subpart) hierarchy, and instances (to separate stereotype or generic objects from their potential realizations in the scene). Instances can be used to explore several possible interpretations without any commitment to their ultimate existence.

VALUEd types provide a slot that may be filled by a value. The slot can have expressions associated with it, such as a default or the required attributes of the slot filler. Furthermore, functions can be initiated when the value is added, modified, removed, or needed. This flexibility is used to maintain the consistency of the interpretation under consideration, and to spread the effects of changes to the appropriate objects. VALUEd types can be used with LINKs to generate property inheritance (usually down a specialization hierarchy). Knowledge sharing of this kind can often compensate for incomplete data.

Procedural attachment is a mechanism which facilitates object-centred control of the interpretation process. In this system, PROCEDUREs are invoked via pattern matching, the primary message-passing utility in Maya. PROCEDUREs will typically exist for situations where the schema is invoked as a model (top-down) or to account for data (bottom-up). This facility

can also be used to invoke procedures appropriate to the context current at the time the schema is entered.

The CONFIDENCE attribute is a number used by the global scheduler to encourage evaluation of the most promising interpretations. Associated with this attribute is a schema-specific algorithm to modify the confidence value whenever VALUEs are found for this object or its components. Using an algorithm provides the flexibility to consider the difficulty and importance of completing the interpretation of an object as well as the probability of its existence.

Schemata are used to represent scene objects in a task domain. Some examples in the geographic domain are roads, rivers, shorelines, bridges, river systems, towns, and road systems. The schemata are combined in decomposition hierarchies (e.g., road systems are made up of roads, bridges, and towns; towns are made up of buildings and roads; and so on) and specialization hierarchies (e.g., lakeshores and coastlines are "types" of shorelines).

The hypothesis of this research is that the combination of two interpretation tasks can work to their mutual benefit. One might assume that this would make the problem more difficult instead of easier. However, the proper mixture of data will resolve ambiguities without increasing inconsistencies. Identification of features in an image, such as roads, rivers, and bridges can be guided by their presence in a sketch map.

3. An Example

A sketch map of Ashcroft, B.C. has been interpreted by Mapsee2 (Havens and Mackworth, 1980; Mackworth and Havens, 1981). The Maya schema which represents the unambiguous representation of a bridge in the scene is in Figure 1.

```
side1:       *chain-3
side1-desc:  ((57 . 33)
              (0.886914 . 0.461934) 54.1202)
side2:       *chain-5
side2-desc:  ((69 . 62)
              (-0.918062 . -0.396436) 47.918062)
regions:     (*region-1 *region-2 *region-3
              *region-4 *region-6)
C/labels:    ((*chain-3 . *bridge)
              (*chain-5 . *bridge))
Q/models:    ((*river-system *river-system-1)
              (*road-system *road-system-1))
```

Figure 1. *bridge-1 an Instance of *bridge

Note that the slot fillers do not differentiate between pointers to other schemata (eg. side1) and numeric values (eg. side1-desc).

A stereotypical bridge schema can also be written in Maids. Figure 2 shows the blank data structure with none of its slots filled.

In addition, there are functions associated with this schema for top-down evaluation, bottom-up evaluation, and to instantiate the bridge in a grey-scale digitized image when it has been identified in the sketch map. The execution of this latter procedure will be followed to indicate how schemata can be manipulated within Maids.

The basis of the routine is the following: Regions in the appropriate area are sought that could be interpreted as road or shadow. Then edges are found in the same area which could

```
sketchmapitem:    value: nil  %confidence: nil

                  %if-added:      (prog nil
              (printlb "value added to sketch
                        map item = " %val))
                  %if-removed:    (prog nil
              (patom "value removed from sketch
                        map item = " %val))
                  %if-modified:   (prog nil
              (patom "value modified in sketch
                        map item = " %val))

orderlist:        value: nil  %confidence: nil
neighbourregions: value: nil  %confidence: nil
shadowregions:    value: nil  %confidence: nil
roadregions:      value: nil  %confidence: nil

a-part-of->       nil      decomposes-to->  nil
instances->       nil      neighbours->     nil

confidence:       nil
conf-alg:
  (prog (lst)
     (setq lst (sgeta %name 'decomposes-to))
     (cond ((null lst)(return 0.0))
           ((atom lst)(return (sgetc lst)))
           (t (return
                (quotient
                  (apply 'plus
                    (mapcar
                      '(lambda (n) (sgetc n))
                      lst))
                  (length 'st))))))
```

Figure 2. The Stereotype %bridge Schema

define the sides of the bridge (and possibly a shadow). While the procedure is interesting by itself, it is shown to illustrate the actual use of the Maids schemata manipulation functions.

The next section consists of brief excerpts of "pseudo-code" (and possibly responses from the program which will be indented and preceded by a ">") followed by an explanation of what is accomplished. The schemata manipulation functions will be underlined throughout. The procedure is entered with the variable SKETCHMAPSCHEMA bound to the "%bridge-1" from Figure 1.

```
| if (SKETCHMAPSCHEMA =
|       (sgeta '%bridge 'sketchmapitem 'yes 'one
|                                  '(instances))
|       (return 'alreadyexists))
```

A check to determine whether this schema has been examined before. Sgeta searches from %bridge down the "instances" links looking for one instance schema containing SKETCHMAPSCHEMA as its sketch map item.

```
| Inst = (snewi '%bridge)
|       > create a new bridge instance: %bridge-1
|
| p1 = (car (sgeta SKETCHMAPSCHEMA 'side1-desc))
| p2 = (car (sgeta SKETCHMAPSCHEMA 'side2-desc))
|       > p1 = (57 . 33)
|       > p2 = (69 . 62)
```

Since Mapsee2 schemata are undifferentiated, get attribute returns the value of slots. P1 and p2 are the locations of the midpoints of the bridge sides.

```
| reglist = (pointstrips p1 p2 %regmatfile 1.5)
|      > regions list = (130 1186 1629 9 1750 2018)
```

Pointstrips searches for all the regions (generated from a region-merging algorithm) in %regmatfile that are enclosed in a rectangular strip whose corners are 1.5 units from p1 and p2.

```
| orderlist = (ordinterp reglist Inst)
|       > order list = (Other Shadow Bridge Bridge
|                       Other)
```

Orderinterp interprets the regions and determines if their interpretations are consistent with regions around and over a bridge. Interpretations include ROAD, WATER, URBAN, SHADOW, MOUNTAIN, and HILLS. The appropriate regions are added to the "%bridge-1" schema in the slots "shadowregions", "roadregions", and "neighbourregions". The order list shows the order of regions from p1 to p2. In this case they are Other, Shadow, and Bridge.

```
| (nputv Inst 'orderlist orderlist)
```

Put the value of orderlist in the VALUEd type of the same name in Inst.

```
| (sputc Inst 'orderlist 100)
```

Put the confidence of orderlist at 100 (the maximum).

```
| edgelist = (pointstrips p1 p2 %edgematfile 1.5)
|      > edges list = (88 11 205 206 241 242)
```

Find all the edge segments in the same rectangular strip that was searched for regions. Edge segments come from groups of zero-crossings of the Laplacian of the Gaussian.

```
| if (null edglist)
|      then (sfail-model Inst)
```

If there are no edge segments then this model can not be correct. So, remove it and any of its descendants from the graph.

```
| if (notwithinrange edgeangle bridgeorientation)
|      then (remove edge edglist)
|      > edges list = (88 11 242)
```

Remove all edge segments that are not similar to the orientation of the bridge in the sketch map (which is 67 degrees).

```
| forall i in edgelist
|    (addto edgelist
|      (lineneighbours i -1.0 35 %edgefile))
|      > edges list = ((88 89)(10 11 12 13 14 15)
|                      (242))
```

Expand all the edge segments in edgelist to include segments out of the rectangular strip that are connected to and in a similar orientation as existing edge segments.

```
| (foreach group of edges i in edgelist
|     (linestats i))
```

Linestats calculates the length of the edge segments, the average contrast across the edge, and the maximum contrast.

```
| (matchregionstoedges orderlist edgelist Inst)
```

Match the regions from regionlist with the edges in edgelist. One result of this is the creation of a new type of schema--the %curb schema. Each %curb schema contains the data concerning one edge of the bridge. A special type of %curb is reserved for shadows. After this routine has executed, the schemata shown in Figure 3 will exist.

```
conf-alg: (prog nil (return
            (quotient
               (add (sgetc %name 'length)
                    (sgetc %name 'maxstrength)
                    (sgetc %name 'angles))
               3)))

confidence:     85

avgstrength: value: 3.5        %confidence  75
maxstrength: value: 5          %confidence  75
length:      value: 20         %confidence 100
angles:      value: (248 214)  %confidence  80
edgesegs:    value: (88 89)    %confidence 100
type:        value: shadow     %confidence 100

a-part-of->     %bridge-1
```

Figure 3a. Instance %curb-1

```
confidence:      93
conf-alg:        (prog nil ...

avgstrength: value: 96.33      %confidence 100
maxstrength: value: 134        %confidence 100
length:      value: 54         %confidence 100
angles:      value: (90 57 79 45 90 27)
                               %confidence  80
edgesegs:    value: (10 11 12 13 14 15)
                               %confidence 100
type:        value: road       %confidence 100

a-part-of->     %bridge-1
```

Figure 3b. Instance %curb-2

```
confidence:      100
conf-alg:        (prog nil ...

avgstrength: value: 22.0       %confidence 100
maxstrength: value: 22         %confidence 100
length:      value: 17         %confidence 100
angles:      value: (228)      %confidence 100
edgesegs:    value: (242)      %confidence 100
type:        value: road       %confidence 100

a-part-of->     %bridge-1
```

Figure 3c. Instance %curb-3

The confidence algorithm is executed after the last value slot is filled by a call to sputc with the appropriate flag set.

```
| riverreg = (sgetv 'rivers 'regions 'yes
|                   'valueonly 'all '(instances))
|      > riverreg = ((%river-1 130 1058 943 874)
|                    (%river-2 1750 2018 2095))
|
| forall regions r in neighbourregions
|     if r is in riverreg
|     then (saddl Inst 'neighbours River))
|      > neighbours = (%river-1 %river-2)
```

Search through all the existing river schemata for the regions they contain. If they match the neighbouring regions of the bridge, add links to the rivers from the bridge and vice versa.

```
| (sputv Inst 'sketchmapitem SKETCHMAPSCHEMA)
| (sputc Inst 'sketchmapitem 100 t)
|      > value added to sketchmapitem = *bridge-1
```

Instantiation has succeeded so add SKETCHMAPSCHEMA as the sketch map item for this schema. Set its confidence to the maximum and propagate that fact. Figure 4 displays the schema for %bridge-1 at this stage.

```
sketchmapitem:      value: *bridge-1
                    %confidence:        100
                    %if-added:      (prog ...
                    %if-removed:    (prog ...
                    %if-modified:   (prog ...

orderlist:          value: (Other Shadow Bridge
                            Bridge Other)
                    %confidence:        100
neighbourregions:   value: (2018 1750 9 130)
                    %confidence:        100
shadowregions:      value: (1186)
                    %confidence:        100
roadregions:        value: (1629)
                    %confidence:        100

a-part-of->         nil
decomposes-to->     (%curb-1 %curb-2 %curb-3)
neighbours->        (%river-1 %river-2)

confidence:         90
conf-alg:           (prog (lst) ...
```

Figure 4. Instance %bridge-1

This is just one way in which the functions in Maids might be used. While only some of its capabilities are shown directly, many of the functions referred to contain embedded calls to other Maids functions. Appendix A lists the complete range of schemata manipulation functions.

4. Discussion

The benefits of cooperating knowledge sources arise because each source deals with similar information in a different way. For example, in sketch map analysis it is often impossible to decide whether the water is inside or outside of a shore. However, the location of the shoreline in the sketch map can guide the search for the appropriate features in the grey-scale image. Then, a simple pixel classification technique applied to the grey-scale image will generally suffice to discover which region corresponds to water. Hence the interpretation of each knowledge source provides useful information to aid in the interpretation of the other source.

A critical aspect of this system is that the schemata can accept and utilize knowledge from cooperating sources of knowledge. This is difficult since the data are usually in very different forms. Even reconciling data from the same source (such as region and edge data) can be problematic. The modularity of the schemata along with a flexible control strategy can be used to overcome this problem.

5. Conclusions

Knowledge sources that can incorporate information from several types of input within the

framework of a world model may be effective in computational vision. This paper describes a schemata-based system that is testing those claims.

References

Bajcsy, R. and Tavakoli, M. (1973) A Computer Recognition of Bridges, Islands, Rivers and Lakes From Satellite Pictures. Proc. Machine Processing of Remotely Sensed Data, Purdue.

Erman, L.D. and Lesser, V.R. (1975) A Multi-Level Organization for Problem Solving using Many, Diverse Cooperating Knowledge Sources. Proc. Fourth IJCAI, Tbilisi, pp. 483-490.

Havens, W.S. (1979) A Procedural Model of Recognition for Machine Perception. TR-78-3, UBC, Vancouver, B.C.

Havens, W.S. and Mackworth, A.K. (1980) Schemata-Based Understanding of Hand-Drawn Sketch Maps. Proc. Third Conf. CSCSI, Victoria, B.C., pp. 172-178.

Horn, B.K.P. and Bachman, B.L. (1978) Using Synthetic Images to Register Real Images with Surface Models. CACM 21, 11, Nov. 78, pp. 914-924.

Koomen, J.A.G.M. (1980) The Interlisp Virtual Machine: A Study of its Design and its Implementation as Multilisp. M.Sc. Thesis, UBC, Vancouver, B.C.

Lesser, V.R. and Erman, L.D. (1977) A Retrospective View of the HEARSAY-II Architecture. Proc. Fifth IJCAI, MIT, pp. 790-800.

Mackworth, A.K. and Havens, W.S. (1981) Structuring Domain Knowledge for Visual Perception. Proc. Seventh IJCAI, Vancouver, B.C., pp. 625-627.

Nitzan, D., Brain, A., and Duda, R. (1977) The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis. Proc. IEEE, 65, Feb. 77, pp. 206-220.

Roberts, R. and Goldstein, I. (1977a) The FRL Manual. AIM 409, MIT.

Roberts, R. and Goldstein, I. (1977b) The FRL Primer. AIM 408, MIT.

Tenenbaum, J.M., Fischler, M.A., and Wolf, H. (1978) A Scene-Analysis Approach to Remote Sensing. TN 173, SRI.

Appendix A. Schemata Manipulation Functions

Schemata

```
(%OBJECT%   VALUED  LINK  PROCEDURE  OTHER)
(%INSTANCE% VALUED  LINK  PROCEDURE  OTHER)

VALUED = attribute (%value S-exp
                    ((%confidence . number)
                     (%default . S-exp)
                     (%if-added . form)
                     (%if-modified . form)
                     (%if-removed . form)
                     (%if-needed . form)
                     (%required . predicate)))
```

Suffix Conventions

A--any Attribute
V--VALUED attributes
L--LINK attributes
I--Instances
O--Objects

Functions

1. (add-link-inverse link inverse)- adds link and its inverse to %link-types.
2. (ifneed schema attribute)- evals the %if-needed clause of attribute in schema.
3. (saddl schema1 link schema2)- adds a link from schema1 to schema2 (and the inverse).
4. (anylink? schema1 schema2)- determines if there is a path from schema1 to schema2 using any type of LINKs.
5. (sattr schema types)- returns a list of the attribute names of schema.
6. (sattrtype schema attribute)-returns the type of attribute in schema.
7. (sconsist)- makes a data base consistent by adding and removing appropriate LINKs and objects.
8. (screate [name])- creates a new stereotype (object) schema.
9. (seraset instance [If-removed] [splice?])- destroys instance.
10. (seraseo object [splice?])- destroys object (after user confirmation).
11. (sgeta schema attribute [inherit allorone links])- gets the value of attribute in schema.
12. (sgetc schema [attribute])- returns the confidence value for a schema or a VALUEd type attribute.
13. (sgetl schema link)- returns the schema(ta) pointed to by link from schema.
14. (sgetv schema attribute [inherit type allorone links1 links2])- gets the value (or default) of attribute (a VALUEd type) in schema.
15. (slink? schema1 schema2 link)- returns the length of the shortest path from schema1 to schema2, if it exists, using only link.
16. (snewi object)-creates a new instance of object and returns its name.
17. (splice schema link [inverse])- removes all LINKs from schema and splices them together.
18. (sprint schema)- pretty prints schema.

19. (sprintn schema link)- pretty prints the tree structure under schema using link.
20. (sputa schema attribute [defn])- puts a new attribute (with value defn) in schema.
21. (sputc schema [attribute cval spread?])-changes the confidence value of the schema or a valued attribute to cval.
22. (sputcl schema attribute value oldconfidence newconfidence spread)-replaces oldconfidence in the list of confidence values of attribute (a VALUED type) in schema with newconfidence.
23. (sputv schema attribute value)-puts a new value for attribute (a VALUED type) in schema.
24. (sremovea schema attribute [If-removed])- removes the definition of attribute from schema.
25. (sremoveal schema link)- removes all LINKs from schema (and the inverses).
26. (sremovel schema1 link schema2)- removes the link from schema1 to schema2 (and the inverse).
27. (sremovev schema attribute)- sets the value of attibute (a VALUED type) in schema to NIL.
28. (srestore [file])- restores all the objects in file.
29. (ssave [file])- saves all objects in file.
30. (ssprintn schema)- pretty prints the graph structure from schema using all LINKs.
31. (vcheck schema attribute value)- checks to see if value meets the conditions in the \required clause in schema.

EXPERIMENTS ON STREAK PREVENTION IN IMAGE RECONSTRUCTION FROM A FEW VIEWS

Richard Gordon and M.R. Rangaraj

Quantitative Morphology Unit
Departments of Pathology and Radiology
Faculty of Medicine, University of Manitoba
Winnipeg R3E 0W3, Canada.

ABSTRACT

Streaks arise in computed tomograms for a variety of reasons such as presence of high contrast edges and objects, aliasing errors and use of a very few views. The problem appears to be an inherent difficulty with all reconstruction methods, including back-projection (with convolution) and the algebraic reconstruction technique (ART). We found that a posteriori removal of streaks is not very effective and can cause further artifacts. This paper presents results of a number of variations of ART oriented towards prevention of streaks. In particular, pattern recognition of streaks during iterative reconstruction and the use of directional neighbourhood operations with ART are considered. Reconstructions of test patterns using the various algorithms derived are presented and optimization of the parameters involved is discussed. As good quality reconstructions are achievable using very few views, our methods can lead to a significant reduction in radiation dose in x-ray computed tomography. They are currently being used for remote computed tomography via teleradiology where a few radiographs acquired at different angles are transmitted to a central computing facility. They should also be useful in electron microscopy of macromolecules and have widespread industrial applications where only a few views are obtainable for non-destructive testing.

INTRODUCTION

Streak-like artifacts arise for a variety of reasons in reconstructions made from projections. There have been a few studies on the causes of such artifacts in computed tomography and correction procedures to remove streaks have also been proposed. For example, Joseph & Spital (1981) describe the "exponential edge-gradient effect" and give Fourier deconvolution and ray sum correction methods to eliminate streaks. The presence of out-of-field objects (Huang et al. 1977) and opaque objects leading to saturated rays (Morin & Raeside, 1981), movement of objects during scanning, and the use of incomplete projection data (Gore & Leeman, 1980; Oppenheim, 1977) have also been pointed out as sources of streaks in reconstructions. Aliasing error caused by inadequate sampling of the projection data is yet another cause of streaking (Brooks et al., 1979; Stockham, 1979). Although the software and mathematics of reconstruction are known to have this inherent weakness (Hall, 1975; Hounsfield, 1977; Moran, 1976), the methods proposed to overcome streaking have only been post-reconstructive cleaning-up procedures (for example, Henrich, 1980) and pre-processing of projection data (for example, Brooks et al., 1979; Joseph & Spital, 1981; Morin & Raeside, 1981). Our initial attempts at processing the reconstructions were in a similar spirit. We used correlation measures for detecting streaks and a variety of linear and nonlinear operators on the streaks, but did not obtain good results: while detection of streaks was easily performed, the choice of parameters for the operators was found to be difficult and picture dependent and the procedure often introduced further artifacts.

We felt that a suitable modification to the reconstruction procedure that would prevent streaks from getting into the reconstruction would be a better approach than trying to remove them after they have already been introduced. This paper presents initial results of variations of ART (Gordon et al., 1970) derived with this approach in mind. In particular, the use of directional neighborhood operations and pattern recognition of streaks during the iterations of an ART algorithm is considered. Reconstructions of test patterns made from a very few views using the new algorithms are presented and optimization of the parameters involved is discussed. As good quality reconstructions are achievable with a very few views, these techniques could lead to a drastic reduction of radiation dose in x-ray computed tomography. They also permit transmission of relatively few radiographic views for remote computed tomography via teleradiology (Rangaraj & Gordon, 1982). Since only a few views can be obtained in many industrial applications of computed tomography, the algorithms should be of considerable use in non-destructive testing. They should also

find applications in electron microscopy (Bender et al., 1970), where electron beam damage of macromolecules limits the number of useful exposures, and thus views, that can be obtained.

THE ALGEBRAIC RECONSTRUCTION TECHNIQUE

The algebraic reconstruction technique (ART) is an iterative procedure which starts with an initial estimate of the picture and updates the pixels so as to satisfy the given projection data. A brief description of ART from a digital picture processing point of view is given below; for more detailed analysis of ART and other reconstruction algorithms, see Gordon et al. (1970) and Gordon (1974).

The image reconstruction problem can be posed as follows: Given a set of projections $R(l,k)$ at angles θ_l, for P views ($l = 1...P$), each view having $2R + 1$ rays ($k = -R...R$), compute a picture $p(i,j)$ such that the raysums $S(l,k)$ of $p(i,j)$ are as close to $R(l,k)$ as possible. The ART algorithm updates pixels belonging to individual rays of a view to meet the raysum criterion as

$$p^{q+1}(i,j) = p^q(i,j) + (R(l,k)-S^q(l,k))/N(l,k) \qquad (1)$$

where $N(l,k)$ is the number of pixels in the ray (l,k) and q refers to the iteration number. The above operation performed over all views constitutes one cycle, and a number of such cycles will have to be executed before all raysums are met. A suitable convergence criterion can be set up based on the error of reconstruction defined as

$$E^q = \sum_l \sum_k |R(l,k) - S^q(l,k)| \; / \; \sum_l \sum_k R(l,k). \qquad (2)$$

The initial picture used is usually a uniformly gray picture of intensity equal to the average brightness of the picture, which may be computed from the given projection data.

Equation (1) represents additive ART, so called because the correction applied is additive. As this can lead to negative values being assigned to pixels, an additional constraint is essential. This is done by setting the pixel to zero whenever the value given by Equation (1) is negative.

The multiplicative version of ART is defined as

$$p^{q+1}(i,j) = p^q(i,j)R(l,k)/S^q(l,k), \qquad (3)$$

which has the advantage that negative pixel values are not encountered.

Six test patterns were used to evaluate ART and the modifications derived. The original test patterns are given in Figures 1a,2a,3a,4a,5a and 6a. (Figures 1a and 5a were acquired using a TV camera and frame buffer; 2a is the negative of 1a; 4a was derived from 2a by setting the background to zero; and, 3a and 6a were created by computation. All patterns are of size 101X101 pixels with a maximum of 256 gray levels). Reconstructions of these patterns by additive ART are given in Figures 1b,2b,4b and 5b; by multiplicative ART in Figures 3b,4c,5c and 6b. Only eight views at angles θ_l = 20,40...140,160 degrees (measured along the rays) were used, with 220 rays per view. Figure 3c is the reconstruction of 3a by multiplicative ART using only four views at 30,70,110 & 150 degrees, included to demonstrate the distortions caused. The raywidth was defined as

$$r_l = \max (|\cos(\theta_l)| , |\sin(\theta_l)|) \qquad (4)$$

such that each ray crosses one and only pixel per row or column. It is seen that the reconstructions have streaks at all projection angles. While the streaks in the additive ART reconstruction extend to the zero background in Figures 4b and 5b, the multiplicative ART reconstructions in Figures 4c and 5c are free of this. This property, however, can be easily incorporated into additive ART by forcing all pixels belonging to rays with zero sums to always remain at zero and excluding them from ART computations (cf. Gordon, 1974). Reconstructions computed with this 'convex hull' feature included in additive ART are given in Figures 4d and 5d. Note that this feature can lead to an improvement only in cases where there are rays with zero sums. In cases of low level background noise, this can be achieved by thresholding projection data. The advantages of using the convex hull are that the raysum distribution is more accurate and some computation is saved by keeping the labeled pixels out of the reconstruction procedure.

FEATURES OF STREAKS

Linear streaks usually occur only along the rays used in reconstruction. The most common cause of streaks is the presence of high contrast edges and objects in the image. The fundamental reason for streaking, however, is that the reconstruction algorithms (both back projection and ART) have a 'smearing' feature: the raysum, or the correction in the case of ART, is spread out uniformly along the path of the ray. Figure 1e gives reconstructions of Figure 1a after each view during the first cycle

of additive ART, where this feature is clearly seen. While this may occur only in the first cycle of ART, the streaks so introduced are usually not corrected by the subsequent iterations. The use of a very large number of views tends to merge the streaks at different angles and give a uniform background. When the number of views used is small, as in our experiments, the streaks remain obvious, as can be readily seen in Figures 1b,2b and 6b.

A study of the streak patterns in Figure 6b points to two types of streaking mechanisms: the smearing of the larger (whiter) raysums leads to the broad streaks which are whiter against the light background (transmission type), and, at the edges of the high contrast objects, dark, tangential streaks arise due to compensations made in the background for whiter values assigned to pixels at the boundaries of the objects (compensatory type). While the transmission type of streaks are easily smoothed out by the use of a large number of views, the compensatory streaks are not.

Treating the pixels of a ray as forming a scan-line signal, we may characterize a streak by a high autocorrelation between successive pixels of the ray. This feature was used in our initial studies to detect streaks. In one procedure we tried, the reconstruction algorithm was made to skip the ray correction if the new values had first and second autocorrelation coefficients higher than a fixed threshold. This procedure, however, failed to prevent streaking. Post-reconstruction processing of streaks so detected by different linear and non-linear contrast stretching operators, with re-normalization to meet the raysum criterion, also failed due to lack of information about the contrast limits of the picture. The result of one such operation, defined as

$$p'(i,j) = \begin{cases} p(i,j)/2 & \text{if } p(i,j) < \text{mean} \\ 2p(i,j) & \text{if } p(i,j) > \text{mean} \end{cases}$$

(where 'mean' is the average value of all pixels along the ray considered) on the streaks in Figure 1b is given in Figure 1c, where it is seen that such a procedure can produce further artifacts (a ray being labeled as streak if the first auto-correlation coefficient of the pixels of the ray exceeded 0.99 or both the first and second coefficients exceeded 0.97).

DIRECTIONAL NEIGHBOURHOOD AVERAGING

As linear streaks ordinarily occur only in the directions of the rays used for scanning, we felt that a suitable criterion based on pixels belonging to adjacent rays, when incorporated into the reconstruction algorithm, would prevent streaking. One such approach is a linear combination of ART as in Equation (1), with a directional neighbourhood averaging which tends to minimize the differences between pixels belonging to adjacent rays. This is achieved by making a pixel value tend to a weighted average of its previous value and those pixels of its 8-neighbourhood that belong to the rays on the left and right of the ray passing through it. Based on the above discussion we define an error measure

$$A[\Sigma\{p^{q+1}(i,j)-(C\ a(i,j) + (1-C)\ p^{q}(i,j))\}^2]$$

$$+ (1-A)[\Sigma\{p^{q+1}(i,j)$$

$$-(p^{q}(i,j)+(R(1,k)-S^{q}(1,k))/N(1,k))\}^2]$$

where the summations are over all pixels (i,j) belonging to ray (1,k). Minimization of this error measure with respect to p(i,j) for iteration q+1, gives the following algorithm:

$$p^{q+1}(i,j) = \max[0, A\{C\ a(i,j) + (1-C)\ p^{q}(i,j)\}$$

$$+(1-A)\{p^{q}(i,j) + (R(1,k)-S^{q}(1,k))/N(1,k)\}]$$

$$0 < A,C < 1. \qquad (5)$$

Equation (5) is the SPART (Streak Preventive ART) algorithm presented in Gordon & Rangaraj (1981). Here, a(i,j) is the average intensity of the immediate neighbours of p(i,j) that belong to rays (1,k-1) and (1,k+1). These pixels are always in the 8-neighbourhood of (i,j) when the ray width is defined as in Equation (4). The weighting factor A determines the proportion of averaging and ART to be used and the factor C controls the weight of the neighbours in directional neighbourhood averaging. As can be seen from Equation (5), A = 0 represents regular ART. The value of the error defined in Equation (2) may be expected to be larger for higher values of A and C: a larger A will make the contribution of ART lesser and a higher C will increase the effect of the neighbouring pixels on the averaging. Further, the values of A and C will together influence convergence of the algorithm. Reconstructions of the patterns in Figures 1a and 2a using Equation (5) are given in Figures 1d and 2c, with A = C = 0.85. It is seen that while streaking is prevented to some extent in Figure 1d, contrast is lost: Figure 1f gives a set of reconstructions of Figure 1a using the SPART algorithm with A (horizontal axis) and C (vertical axis) at 0.1, 0.35, 0.6 and 0.85. Smaller values of A and C were observed to lead to reconstructions with streaks prevented to a lesser extent, but with higher contrast.

The SPART algorithm failed in the case of Figure 2c. This indicates that a uniform

smoothing is not desirable. Making C a function of the local contrast, as

$$C = |a(i,j)-p(i,j)|/(a(i,j)+p(i,j))$$

yielded the reconstruction in Figure 2d, which does not show much improvement. In an attempt to recover contrast, we gradually reduced the weighting factor A for successive iterations down to zero, which resulted in the reconstruction in Figure 2e. An interesting observation from this case is that streaks arise even if a better estimate than a flat picture (in this case a blurred reconstruction) is given to ART.

A point of note about the smoothing performed by the SPART algorithm is this: the operation is directional, being perpendicular to the ray considered. Thus post-reconstruction blurring will not lead to same results, with standard image processing operators. Further, the algorithm attempts to prevent streaks, rather than just blur them by averaging. This is demonstrated in Figure 1g where results of application of the digital Laplacian operator (see Rosenfeld & Kak, 1976) to Figures 1a,b and d are given.

OPTIMIZATION OF NEIGHBORHOOD OPERATIONS

From the results obtained from experiments discussed in the previous section, it is apparent that the neighborhood operations must be more flexible and picture dependent. A different approach, one of optimization of a given cost function subject to a constraint, was taken. Let a(i,j) denote a linear combination of the immediate neighbors of p(i,j) which belong to the rays to the left and right of the ray passing through p(i,j). In order to prevent occurrence of streaks, we may proceed to iteratively minimize

$$\Sigma \{ p(i,j) - a(i,j) \}^2$$

subject to the constraint

$$\Sigma p(i,j) = R(l,k)$$

where the summations are over all pixels belonging to ray (l,k). Applying Lagrange's method of undetermined multipliers (see, for example, Burley, 1974) we get the following algorithm:

$$p^{q+1}(i,j) = a(i,j) + (R(l,k) - \Sigma a(i,j))/N(l,k) \qquad (6)$$

where the summation is over all a(i,j) associated with ray (l,k). The multiplicative version of the above equation can be expressed as

$$p^{q+1}(i,j) = a(i,j)R(l,k)/\Sigma a(i,j). \qquad (7)$$

The important feature of the above algorithm is that it tends to keep the difference between adjacent ray scan-lines to a minimum, satisfying the raysum constraint at the same time. This represents an improvement over SPART.

Initially we defined a(i,j) as the average of all neighbors of p(i,j) belonging to rays (l,k-1) and (l,k+1). Reconstructions using the additive version in Equation (6) are given in Figures 4e and 5e (with the convex hull feature described earlier); Figures 4f and 6c are reconstructions using the multiplicative version in Equation (7). In an attempt to improve the contrast in these reconstructions, the geometric mean of the expressions in Equations (3) and (7) was tried, defined as

$$p^{q+1}(i,j) = \text{sqrt}[\{a(i,j)R(l,k)/\Sigma a(i,j)\} \{p^q(i,j)R(l,k)/S^q(l,k)\}]. \qquad (8)$$

Reconstruction of the test pattern in Figure 4a using this version is given in Figure 4g. Notice from the reconstructions in Figures 4e,f and 5e that the last projection used (at 160 degrees here) leaves a patchy appearance in that direction. To overcome this "last view effect", the reconstructions obtained at the end of each view were stored and then averaged during the final iteration: the reconstructions so obtained are given in Figures 4h and 5f.

In order to avoid loss of details, the neighborhood factor a(i,j) has to be appropriately defined at each location using the information available at that cycle in such a way as to prevent both streaking and indiscriminate blurring of features. With these considerations in mind we defined three contrast measures to determine a(i,j):

$$\text{along} = (|p_-(i,j)-p(i,j)| + |p_+(i,j)-p(i,j)|)/(p_-(i,j)+p_+(i,j)+2p(i,j))$$

$$\text{across} = (|l(i,j)-p(i,j)| + |r(i,j)-p(i,j)|)/(l(i,j)+r(i,j)+2p(i,j))$$

$$\text{left-to-right} = |l(i,j)-r(i,j)|/(l(i,j)+r(i,j))$$

where p(i,j) is a pixel belonging to ray(l,k), $p_-(i,j)$ and $p_+(i,j)$ are neighbors of p(i,j) along the ray scan-line (l,k), l(i,j) is the average of neighbors of p(i,j) belonging to ray

(1,k-1), and r(i,j) is the average of neighbors of p(i,j) belonging to ray (1,k+1). Note that with the above definitions, the contrast measures have a convenient range of 0 to 1. An important feature simplifies computation: the pixels used are in the 8-neighborhood of p(i,j) when the raywidth is defined as in Equation (4). Three options were provided in the definition of a(i,j) using the above contrast measures:

```
if (across > lim1) and (along < lim2)
 1. then { streak }
  if left-to-right > lim3
   1a. then { edge }
    a(i,j)=nearer of l(i,j) and r(i,j) to p(i,j)
   1b. else { hump }
    a(i,j) = 0.5(l(i,j)+r(i,j))
 2. else { proceed with ART }
  a(i,j) = p^q(i,j)
```

where lim1, lim2 and lim3 are thresholds defined for the three contrast measures to detect streaks and edges in the reconstructed picture; 'edge' refers to a large difference across the pixel as at the edge of an object; and 'hump' refers to a streak over a flat background, in which case the middle pixel would be higher (or lower) than both the left and right neighbors. By setting the contrast thresholds to the maximum possible value (1.0), the algorithm may be forced to follow regular ART. To take maximum advantage of the above operations, the initial picture for ART was computed from the first view data only (by giving to each pixel the value R(1,k)/N(1,k) appropriately), and the algorithm was started with the second view. Figure 6d gives the reconstruction of 6a using the above criteria with renormalization using Equation (7) (with lim1 = lim2 = 0.05, lim3 = 0.2). The dark streaks seen in Figure 6b are not present in Figure 6d. Further, the background is more uniform and the edges of the objects are sharper than in Figure 6c. An optimized choice of the thresholds should lead to an even better reconstruction.

DISCUSSION

When we use only a few views in reconstruction from projections, we are dealing with highly underdetermined equations (Gordon, 1974). For example, a 100x100 array with 8 views represents 10,000 unknowns with around 800 equations (of the form $R(1,k) = \Sigma\, p(i,j)$). Standard algorithms, such as ART, select but one of the infinite number of possible solutions. What we are attempting to do is find alternative solutions to the equations (cf. Gordon, 1973) which are free of the streaking artifacts typical of ART and other algorithms. The algorithms derived have demonstrated their ability to select better solutions to the same set of equations. The use of pattern recognition and all available information about the image being reconstructed to build some intelligence into the algorithm would make selection from the set of infinite solutions easier. Our work has been progressing in this direction and methods of improving performance of the algorithms derived are being researched. The convergence properties and effects of noise will also be studied at a later stage.

ACKNOWLEDGEMENTS

This work was supported by a Strategic Grant from the Natural Sciences and Engineering Research Council of Canada and by a grant from the Control Data Corporation. We thank Messrs. John Dunning, Ron Laramee, and William Paley for constructing the image processing system with which this work was done.

REFERENCES

Bender, R., Bellman, S.H. & Gordon, R., 1970, ART and the ribosome: A preliminary report on the three dimensional structure of individual ribosomes determined by an algebraic reconstruction technique, J. Theor. Biol., 29, 483-488.

Brooks, R.A., Glover, G.H., Talbert, A.J., Eisner, R.L. & DiBianca, F.A., 1979, Aliasing: A source of streaks in computed tomograms, J. Comput. Assist. Tomogr., 3, 511-518.

Burley, D.M., 1974, Studies in optimization, Wiley, N.Y., Chapter 1.

Duerinckx, A.J. & Macovski, A., 1979, Nonlinear polychromatic and noise effects in x-ray computed tomography images, J. Comput. Assist. Tomogr., 3, 519-526.

Gordon, R., Bender, R. & Herman, G.T., 1970, Algebraic reconstruction techniques (ART) for three dimensional electron microscopy and x-ray photography, J. Theor. Biol., 29, 471-481.

Gordon R., 1973, Artifacts in reconstructions made from a few projections, Proc. First Intl. Jt. Conf. on Patt. Recog., Oct. 30 - Nov. 1, Washington D.C., IEEE, 275-285.

Gordon R., 1974, A tutorial on ART (Algebraic Reconstruction Techniques), IEEE Trans. Nucl. Sc., NS-21, 78-93.

Gordon R. & Herman, G.T., 1974, Three dimensional reconstruction from projections: A review of algorithms, Int. Rev. Cytol., 38, 111-151.

Gordon, R. & Rangaraj, M.R., 1981, The need for cross-fertilization between the fields of profile inversion and computed tomography, Proc.

7th Canadian Symposium on Remote Sensing, Winnipeg, September 8-11, 1981, in press.

Gore, J.C. & Leeman, S., 1980, The reconstruction of objects from incomplete projections, Phys. Med. Biol., 25, 129-136.

Hall, E.L., Huth, G.C., Gans, R.A. & Reed, I.S., 1975, Artifacts in three dimensional reconstruction from medical radiographic data, Proc. 4th Intl. Congr. for stereology, 4-9 Sept. 1975, Gaithersburg, Maryland.

Henrich, G., 1980, A simple computational method for reducing streak artifacts in CT images, Comput. Tomogr., 4, 67-71.

Hounsfield, G.N., 1977, Some practical problems in computerized tomography scanning, in: Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine, Eds. M.M. Ter-Pogossian et al., (Baltimore: University Park Press), 217-224.

Huang, S.C., Phelps, M.E. & Hoffman, E.J., 1977, Effect of out-of-field objects in transaxial reconstruction tomography, in: Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine, Eds. M.M. Ter-Pogossian et al., (Baltimore: University Park Press), 185-198.

Huesman, R.H., 1977, The effects of a finite number of projection angles and finite lateral sampling of projections on the propagation of statistical errors in transverse section reconstruction, Phys. Med. Biol., 22, 511-521.

Joseph, P.M. & Spital, R.D., 1978, A method for correcting bone induced artifacts in computed tomography scanners, J. Comput. Assist. Tomogr., 2, 100-108.

Joseph, P.M. & Spital, R.D., 1981, The exponential edge-gradient effect in x-ray computed tomography, Phys. Med. Biol., 26, 473-487.

Moran, P.R., Barroilhet, L.E. & Witt, R.M., 1976, Qualitative and quantitative artifact from CT algorithms, Proc. SPIE, Vol. 96, 138-142.

Morin, R.L. & Raeside, D.E., 1981, A pattern recognition method for the removal of streaking artifact in computed tomography, Radiol., 141, 229-233.

Oppenheim, B.E., 1977, Reconstruction tomography from incomplete projections, in: Reconstructive Tomography in Diagnostic Radiology and Nuclear Medicine, Eds. M.M. Ter-Pogossian et al., (Baltimore: University Park Press), 155-184.

Rangaraj, M.R. & Gordon, R., 1982, Computed Tomography for remote areas via Teleradiology, Proc. SPIE, vol. 318, 182-185.

Rosenfeld, A. & Kak, A.C, 1976, Digital picture processing, Academic Press, New York, Chapter 6.

Shepp, L.A. & Stein, J.A., 1977, Simulated reconstruction artifacts in computerized x-ray tomography, in: Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine, Eds. M.M. Ter-Pogossian et al., (Baltimore: University Park Press),33-48.

Stockham, C.D., 1979, A simulation study of aliasing in computed tomography, Radiol., 136, 721-726.

Tofts, P.S. & Gore, J.C., 1980, Some sources of artefact in computed tomography, Phys. Med. Biol., 25, 117-127.

Zatz, L.M., 1977, The EMI scanner: Collimator design, polychromatic artifacts and selective material imaging, in: Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine, Eds. M.M. Ter-Pogossian et al., (Baltimore: University Park Press), 245-266.

FIGURE CAPTIONS

1. a)Pattern 1; b) Additive ART reconstruction; c) Streak processing applied to b); d) SPART reconstruction (A=C=0.85).

1. e) View-by-view results of ART reconstruction as in Figure 1b.

1. f) SPART reconstructions of 1a for A (horizontal axis) and C (vertical axis) at 0.1, 0.35, 0.6 and 0.85.

1. g) Results of application of digital Laplacian operator to 1a,b and d. Note absence of streaks in 1d.

2. a) Pattern 2; b) Additive ART; c) SPART reconstruction (A=C=0.85); d) SPART with variable C; e) SPART with sliding A.

3. a) Pattern 3; b) Multiplicative ART, 8 views; c) Multiplicative ART, 4 views.

4. a) Pattern 4; b) Additive ART; c) Multiplicative ART; d) Additive ART with convex hull; e) Reconstruction by Equation (6); f) Reconstruction by Equation (7); g) Reconstruction by Equation (8); h) Same as (g) but with averaging during last cycle.

5. a) Pattern 5; b) Additive ART; c) Multiplicative ART; d) Additive ART with convex hull; e) Reconstruction by Equation (6); f) Reconstruction by Equation (8) with averaging during last cycle.

6. a) Pattern 6; b) Multiplicative ART; c) Reconstruction by Equation (7) with averaging during last cycle; d) Reconstruction using Equation (7) with the three options for a(i,j).

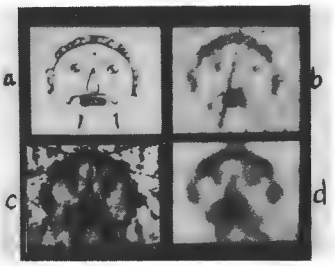


Figure 1a-d

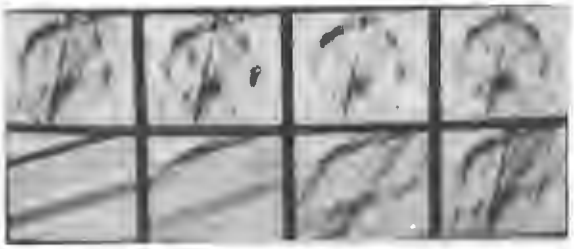


Figure 1e

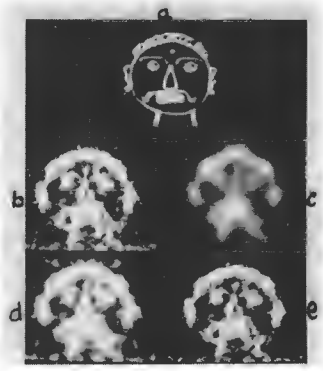


Figure 2 a-e

Figure 3a-c

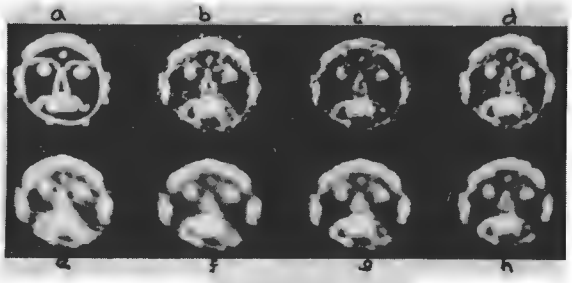


Figure 4 a-h

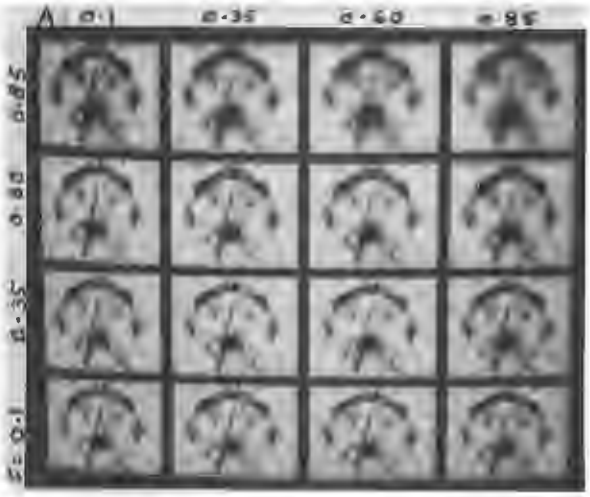


Figure 1f

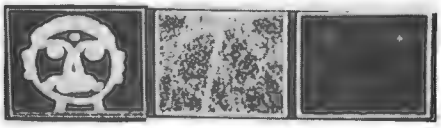


Figure 1g

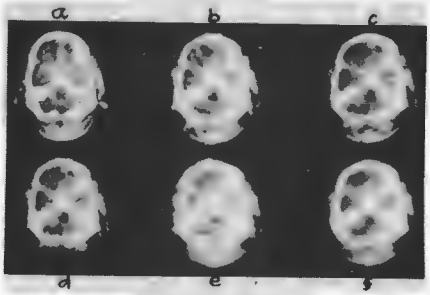


Figure 5a-f

Figure 6a-d

Feature Constraints for Computer Interpretation of CAT Scan Images of Logs

Edwin Bryant and Brian Funt

Department of Computing Science
Simon Fraser University

Abstract

Computerized Axial Tomography (CAT) scans of logs reveal details of their internal structure. We have determined features which can be extracted and used as constraints in a computer program to interpret CAT scan images of logs.

Introduction

A log entering a sawmill is seldom perfect. It will have knots or rot inside it, and it is not possible to ascertain where these imperfections are buried by simple visual inspection of its surface. When the log is cut the knots and rot become obvious, but by then it is too late. If the imperfections in a log could be found before cutting it, then it would be possible to cut around them. Industrial tomography has been shown to be a viable method for non-destructive investigation of the internal structure of objects (Ellinger 1979, Hopkins 1981, Reimers 1980). In particular, tomography of logs has shown that defects and other aspects of internal structure are readily discernible.

This paper describes a set of features by which CAT scan images of log cross-sections can be segmented into regions of knots, rot, clear wood etcetera. These regions are identified by features such as the length, strength, and direction of growth rings, the density of the wood, and the subjective contours formed where a set of growth rings is interrupted by a hole or a rotten region. Following Marr (Marr 1978), Barrow and Tenenbaum (Barrow 1978), and others we have looked to the real-world domain to provide constraints which limit the possible interpretations of image features. First, the CAT scanning process, and the visible features in a log cross-section will be described. Next, the processing methodology will be described, including primary feature extraction and, at a higher level, segmentation processing.

CAT Scans and Logs

Computerized Axial Tomography is a well-developed medical imaging technique which is currently being applied in non-medical areas. CAT scanning is a method whereby a set of x-rays of a given object is used to create a cross-sectional density map. The output of this process is a gray scale image, typically displayed on a TV monitor, with the intensity value of any pixel corresponding to the density of that area of the scanned object.

CAT scans of logs exhibit readily discernible features even to the untrained observer. Figure 1 is a diagram of the features which can be seen in a typical CAT scan of a log. These include:

1. growth rings-- distinguished by their high density and circular shape about the center of growth,
2. holes, cracks, pitch pockets-- distinguished by their low density (they appear as light regions in actual CAT scans),
3. knots-- distinguished by their high density, roughly elliptical shape (major axis pointing towards the center of growth), and the fact that knots tend to distort the growth rings near them;
4. rot-- distinguished by its higher, lower, or varied density, and the way rotten areas interrupt growth rings.

Processing Methodology

Although the input to this system is from a limited domain, the images themselves can vary greatly. Logs vary across species. Also the water content of a log affects its CAT scan image. Techniques such as statistical pattern recognition, or segmentation-by-thresholding will not yield consistent results in this environment. They can, however, be used at lower levels to extract simple features. Knowledge-directed interpretation of the input image is needed at higher levels of processing. The image segmentation process divides nicely into two phases. The first is to extract primary or simple features. The second is to use these primary features to determine the image regions corresponding to the internal structure of the log.

a. Primary Feature Extraction

Simple thresholding suffices to determine the external boundary of the log since the density difference

between air and wood is so pronounced.

A standard edge-detection operator (Marr 1979) provides a set of edge points. Clearly, growth rings generate edge elements, as do cracks, holes, and knots.

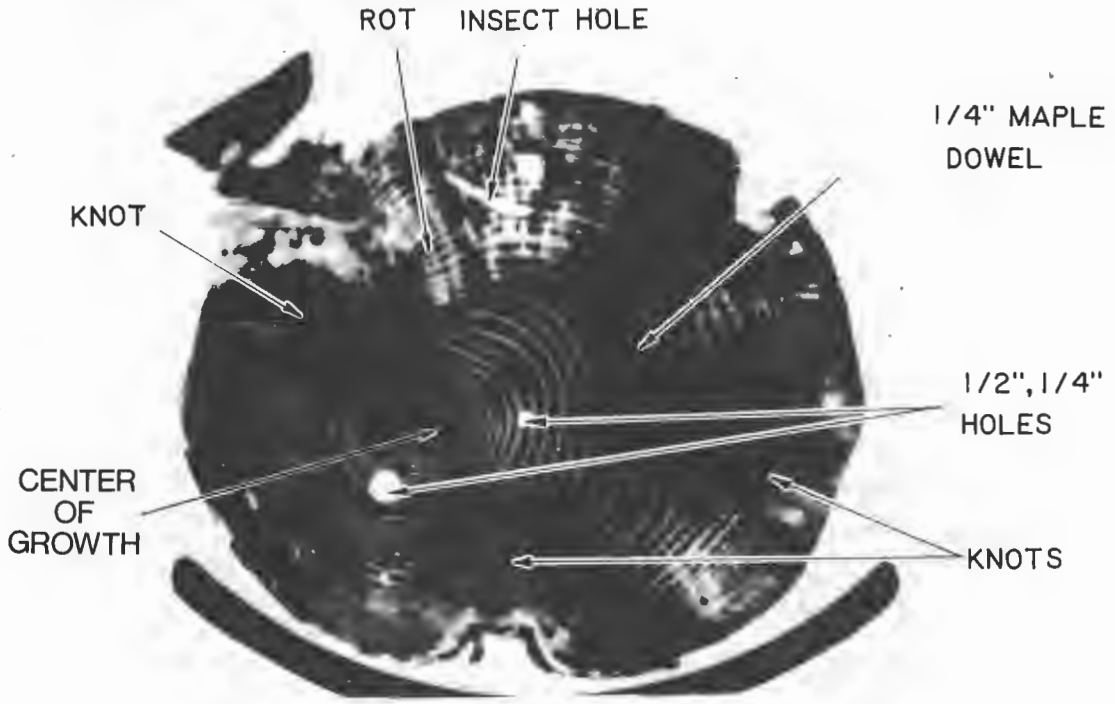


Figure 1
CAT scan image of a log.

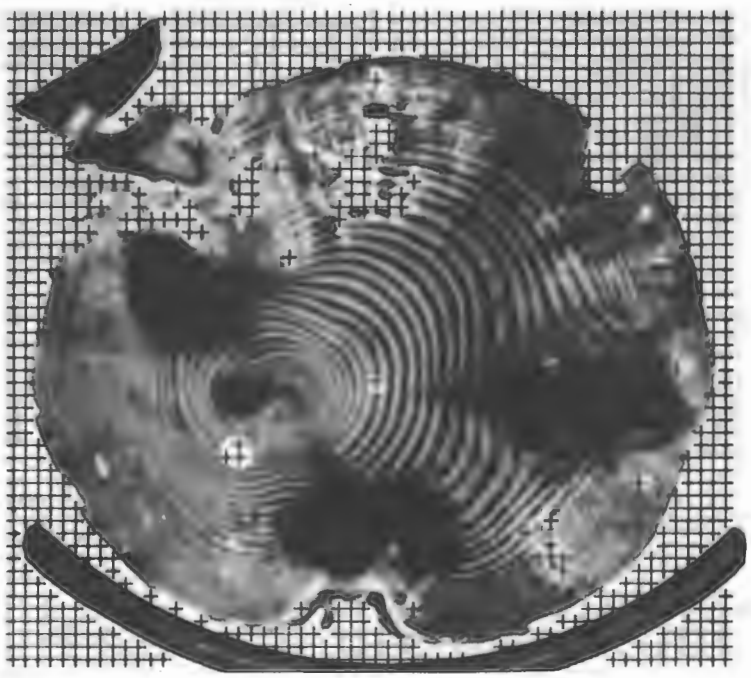


Figure 2
The cross-hatched areas correspond to regions of low density superimposed on the original image.

b. High-level Features

The primary features available are the original density image, the external boundary of the log, and a set of edge elements. The classification of image regions depends in large part upon the global context provided by the growth rings. This context dependency is not arbitrary, for growth rings are a strong visual cue for human observers. Neighbouring edge elements having similar characteristics are linked together into curved line segments. Those which are concentric with the log boundary are generally growth rings. Since we know approximately where the growth ring segments should continue we can try to extend them in a manner similar to that which Shirai (Shirai 1975) used for finding lines in the blocks world.

Another strong visual cue is the center of growth of the log. The bias of the center of growth away from the geometric center of the log often causes weak and closely packed growth rings. The center of growth is defined by the center of the set of concentric growth rings. Any two areas of obvious, unperturbed growth-- as indicated by uniform growth rings-- can be used to locate the approximate center of growth. It is located where the perpendiculars to the growth rings in each area intersect. This estimate of the location of the center of growth can be improved upon by repeating the same process on nearby uniform growth areas which are closer to the center.

The primary features and the center of growth are used to define and describe areas of clear wood, knots, and rot. Concentric and uniform growth rings are strong evidence of clear wood. There may be other regions of clear wood, but indicated by weaker evidence. Slow tree growth and limited image resolution cause some areas of clear wood to appear to be of almost uniform density. A region is interpreted as clear wood if its density is uniform and similar to that of established areas of clear wood.

Regions which cannot be classified as clear wood must contain some defect which causes the growth rings to become discontinuous. Our aim is to classify these regions as knots, rot, cracks, or holes. The boundary between these regions and clear wood is a subjective contour formed by breaks in the growth rings.

Many characteristics indicate that a region is a knot. Knots are of high density, and roughly elliptical. The major axis of this ellipse points toward the center of growth. There is usually a 'tail' of high density (see Figure 1) pointing in the direction of the center of growth. Knots also tend to perturb the growth rings near them. Thus growth rings in the image which bend out of the circular pattern are evidence that they are near the boundary of a knot.

Rot causes a log's cellular structure to decay into a mass similar to packed sawdust. Since rot occurs after the growth rings are established, they do not bend out of the circular pattern near the boundary of the rot. The water content within a rotten area may vary as well, causing regions of higher or lower density in the CAT scan image (see Figure 1). Generally, rot is not found in small pockets embedded in clear wood, but rather in extensive areas, often a complete quadrant of the log. All this means that in CAT scans a region of rot can be identified by the presence of fragmented growth rings and widely varying density.

Cracks and holes are usually small regions. They are typically bounded on all sides by clear wood. They also exhibit a uniformly low density throughout. Holes are typically circular in nature, and cracks run radially inward toward the center of growth.

Computer Processing

Some results of the low-level feature detection operations are shown in Figures 2 and 3. Simple thresholding distinguishes low-density regions corresponding to the external boundary of the log, and holes, cracks, and some rot. Figure 2 shows the CAT scan of the log and the thresholded area. Figure 3 shows the edge points resulting from the application of an edge detector. The growth rings in areas of sound wood and the boundaries of knots are readily discernible. In areas of rot, the edge points indicate regions where the density is rapidly changing. The fact that there is no uniform direction to these edge points is one indicator of rotten wood.

Summary and Future Work

Knowledge of how the internal structure of a log is formed constrains the interpretation of CAT scan image features. A log's boundary and growth rings provide a context for the interpretation of local image regions. Preliminary computer work indicates that

the internal structural features of a CAT scan of a log can be readily found. Our current work is to incorporate this knowledge into a computer program that will classify local image regions of CAT scans of logs.

Acknowledgement

The financial support of NSERC grant A4322 is gratefully acknowledged. We also thank Paul Tonner of Atomic Energy Canada Limited, and Dr. S. L. Fransman of Kingston General Hospital for their assistance in obtaining CAT scan data. In addition, we are indebted to the University of Wisconsin at Madison Visionlab for the use of their image processing equipment.

Bibliography

Barrow, H. G., and Tenenbaum, J. M., Recovering Intrinsic Scene Characteristics from Images, Computer Vision Systems, Hanson, A. R. and Riseman, E. M., eds., Academic Press, New York, 1978.

Ellinger, H., Morgan, I. L., Klinksiek, R., Hopkins, F., Tomographic Analysis of Structural Materials, Proceedings of the Society of Photo-Optical Instrumentation Engineers, vol. 182, 1979, pp. 179-186.

Hopkins, F. F., Morgan, I. L., Ellinger, H. D., Klinksiek, R. V., Meyer, G. A., Thompson, J. N., Industrial Tomography Applications, IEEE Transactions on Nuclear Science, vol. NS-28, no. 2, 1981.

Marr, D., Representing Visual Information, Computer Vision Systems, Hanson, A. R., Riseman, E. M., eds., Academic Press, New York, 1978.

Marr, D., and Hildreth, E., Theory of Edge Detection, A. I. Memo 518, Massachusetts Institute of Technology, 1979.

Reimers, P., Heidt, H., Stade, J., Weise, H., Applications of Computerized Axial Tomography to the Field of Nondestructive Testing, Materialpruf (Material Testing - Germany), vol. 22, no. 5, 1980.

Shirai, Y., Analysing Intensity Arrays using Knowledge about Scenes, The Psychology of Computer Vision, Winston, P. H., ed., McGraw-Hill, New York, 1975.

Figure 3
Edge elements resulting from application of the edge detector.

SETTING STRATEGY IN A RULE BASED VISION SYSTEM

Ahmed M. Nazif and Martin D. Levine

Computer Vision and Graphics Laboratory
Department of Electrical Engineering
McGill University

ABSTRACT

The strategy of a rule based computer vision system is defined and analysed. The first aspect involves determining the spatial order in which the system will process an image. In addition, an ordering has to be established among all the rules included in the model. A global strategy is achieved by dividing an image into areas of attention. These are processed in an order that depends on their properties, as reflected by a set of performance measures, computed for each area. A local strategy is then dynamically evaluated within each area, to determine the order of analysis of the regions and lines, as well as the order of application of the rules to the data. Dynamic strategy setting is formulated as a fuzzy decision-making problem, whose solution depends on the performance parameters for each of the areas in the image.

ACKNOWLEDGEMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council under Grant A4156, and in part by an FCAC grant awarded by the Department of Education, Province of Quebec, under Grant EQ-633.

1. INTRODUCTION

Setting strategy is an important factor in the design of a system, since it bears a direct consequence on determining how it will function. For a rule based system, this involves the fundamental issue of determining the order in which the rules will be matched against the data. In addition, for systems that deal with complex data structures, as is the case with a computer vision system, an ordering must be imposed on the data to be matched.

For the rule based image segmentation system described in [Levine and Nazif, 1982 a] both rule ordering and data ordering are part of the system strategy. The knowledge rules are matched against the data stored in a short term memory (STM). This, therefore, requires the system to know the order in which the rules are to be matched. This is especially important, when the data are such that more than one rule can fire.

The data are composed of areas in the image that contain the regions and lines defining a segmentation for that image. Once the order of the rules is established, the system will start a rule cycle by matching the conditions of a specific rule to specific data entries. These conditions require that the features of a region, line or area be tested for the presence of a certain data configuration. Thus, the system must know the specific region, line and area for which this is to be done at any point in processing. Conceptually, the system is searching the image for the occurrence of the data configuration represented by a rule. It is the order of that search that must be established here. This translates to the problem of selecting which area of attention in the image to work on at any point in processing, and specifying the order, within this area, in which the regions and lines are to be visited.

Image processing systems in the past have not seriously addressed the problem of determining the order in which an image is to be processed. However, there is ample proof that such an ordering affects the results of the analysis. This is evident, for example, in region growing systems, in which initial regions are merged in a pre-specified order to form larger regions. It has been shown [Nazif and Levine, 1981] that changing the order in which the regions are merged will alter the configuration of the resulting regions in the final segmentation. This is true irrespective of the fact that the merging criteria remain unchanged. Thus, this aspect of setting strategy is an important one, and in fact, it is responsible for a lot of the errors in the results of segmenting natural scenes using existing segmentation systems.

The inclusion of focus of attention areas in the data structure is significant [Levine and Nazif, 1982 a]. These areas guide the system to parts of the image that should be processed first. The system strategy establishes an order for the areas, as well as within the areas. Furthermore, that order is not a fixed one. It varies from one area to the next, according to the properties of the data within each. We thus introduce the notion of a dynamic data ordering in processing an image. The implementation of such an ordering will be described here in detail as part of the system strategy.

In a more general sense, the strategy for this system includes not only specifying the order of data processing, but also the order in

which the processing criteria embodied in the rules is applied to the data. Unlike previous rule based systems, this order will be shown to be a dynamic one as well. The system can actually vary the order in which it matches its rules on the data, based on the characteristics of the data themselves. This idea of a dynamic data driven rule ordering methodology is also a new one, and helps improve both the efficiency of computation and the quality of the output.

In the next sections, we will define how the strategy for the rule based segmentation system is determined. To do this, we first define what the elements that compose a strategy are. We then formulate the problem of strategy selection as that of a search within the space composed by these elements, for the most appropriate strategy to use for each data configuration. To solve this dynamic search problem, we define performance parameters that are evaluated by the system at different parts of the image. They are used to determine the elements of strategy for each of these parts individually. We will also introduce the constraints that define the relation between the strategy elements and each of the performance parameters. The framework for the solution is shown to be that of a decision-making process, in which decision functions are evaluated from the performance parameters, and are then used to instantiate the system strategy elements. The concepts of fuzzy set theory serve as the design tool for this process. Fuzzy logic will be used in representing the constraints, defining the objectives, and designing the decision-making algorithm.

2. STRATEGY DEFINITION

It is seen from the above argument that in order to define a strategy for the image processing rule based system, we need to answer two basic questions:

(1) Where (in the image) do we go to next?

(2) What do we do when we get there?

The answer to the first question provides the means for selecting the next data entry in the STM to match the rules on. Within the context of the low level processing system described here, two levels of data selection are indicated. A given strategy should first detail the global method by which the next area in the image is to be selected for processing. It should then proceed to specify the criteria for locally selecting the next region and the next line in that area to be processed.

The answer to the second question also constitutes part of the local strategy. After the next data entry to match on has been selected, be it a region or a line, there still remains more than one alternative. These correspond to the different options provided by having more than one knowledge source to instantiate, or in other words, more than one rule to match. Would it be better to match REGION or LINE rules first? Do we give priority to merging over splitting or should we do the opposite? For lines, do we first add new lines or delete existing ones? When do we move on to the next region or line? In general, given a certain data configuration, more than one rule can match, and the question is: if this happens, which rule should fire? For rule based systems this is known as the conflict resolution problem. The local strategy within an area should also provide the solution to this problem by specifying an ordering on the rules to be matched. This ordering will thus vary from one area to another, depending on the features that describe each area. We will now proceed to define the local strategy more precisely by analysing its basic elements.

2.1 Local Strategy Elements

The information required in order to specify the system behaviour within an area of attention is organized into six basic elements. The local strategy can thus be described by a state vector with six components, that are referred to as the "elements" of the local strategy. These are selected to be independent entities, that completely address the strategic aspects of the problem. Each element can assume specific "states" that are attached to it by the system. Following is a discussion of each element.

2.1.1 Region Path Strategy

This element of local strategy determines which region to address next after processing the current region. Two possibilities exist: either a random selection of another region in the current area of attention, or a selection based on the features of the current and new regions. The FOCUS OF ATTENTION process is the one responsible for obtaining the next region. It does so by matching its own rules against the data in the STM. It is the action of these rules that determines the method by which the next region is to be fetched. Different actions are provided by the FOCUS OF ATTENTION process. These correspond to the different states of the region path strategy element. They include getting the region with highest or lowest adjacency value to the current region; getting the adjacent region with the highest or lowest area; getting the region that is encountered when scanning the image in a raster scan, starting at the current region; or getting the region with the next highest region label. A specification of the region path strategy is thus equivalent to the selection of one of the above options.

2.1.2. Line Path Strategy

The determination of which line comes next is the same as in the case of regions, except for the fact that the features used are different. The FOCUS OF ATTENTION process can invoke one of the following actions: getting the closest line that is in front of, behind, or parallel to the current line; getting the longest or shortest line that is near the current line; getting the

line with highest or lowest gradient that is near* the current line; getting the next line in a raster scan; or getting the line with the next highest label. When there is more than one line in front (behind or parallel), the first three options will get the one with the lowest distance from the current line. The next four options, on the other hand, will get the line with a specific feature, with respect to the other lines.

2.1.3 Sequential/Parallel Strategy

When a region is visited by the system, the REGION ANALYSER matches it's rules on the features of this region. If all the conditions of a rule match, the rule will fire, and an action will be executed. This will result in the modification of the current region. Once this is done, the system has two recourses. It may again match the REGION rules to the same region, so that the same rule, or others, may fire and further modify this region. This can be repeated until no more REGION rules match their conditions on the features of the current region. Alternatively, the system can move on to the next region, so that it will not visit the same region twice until all the other regions in the same area have been visited at least once each. The first method produces a sequential order of processing, whereas the second simulates parallel processing.

2.1.4 Processor Priority

One of the questions that arises during processing is that of whether to process regions or lines. The system is faced with this choice at the end of every rule cycle. The process that executes the choice is the SCHEDULER, whose meta-rules can specify one of two actions: match REGION rules, or match LINE rules.

2.1.5. Rule Priority

As with all rule based systems, the order in which the rules are matched constitutes an important part of the system design. This affects both its behaviour and its performance. With a rigid design, rule ordering is embedded in the structure of the system itself. A change in that ordering is virtually impossible without major modifications. This is not the case in production systems, where the production rules are kept separate from the processing modules of the system. Therefore, an explicit rule ordering must be specified in order to carry out the rule matching process. This is usually done by attaching priorities to the rules, so that in case of conflict, the rule with higher priority will fire. This priority can be explicit, or it can be implicit by the temporal order in which the rules are matched. In both cases, the ordering specified is static and cannot be changed during processing. In fact, all rule based systems developed so far using the production system approach have static conflict resolution methodologies.

The system described here implements a dynamic strategy approach which implies a dynamic change in the ordering of the rules. The system knowledge rules are classified into sets that include REGION, LINE, and AREA analysis rules. We observe that the previous strategy element, that of process priority, does in fact establish an ordering on the rules. It specifies whether first to match REGION rules or LINE rules. Since this is a dynamic process that changes from one area in the image to the next, this results in a dynamic ordering of the rules sets.

We now carry the above argument one step further, in order to establish an ordering within each of the rule sets. Two strategy elements result from imposing such an ordering, they are: REGION rule priority, and LINE rule priority.

2.1.5.1. Region Rule Priority

One way to order the rules is by their actions. An ordering is imposed so that rules with certain actions have priority over rules with others. For REGION rules, two types of actions are possible, merging and splitting. Depending on the performance parameters that describe a certain area, the system may impose a priority for merging over splitting in that area, or vice versa. A dynamic ordering of the rules by their actions is thus established.

A further ordering of rules with the same type of action is possible. This is done by using the conditions associated with the rules to specify priorities for rules having the same action. One may argue that rules with a larger number of conditions should have priority over those with a smaller number. In the system described here, conditions that take into account both region and line information take priority over those that consider only one or the other. Unlike the action based ordering, this condition based ordering is a static one, that can be reflected in the order in which the rules are matched.

Rules with the same type of action are sorted according to their established priorities prior to the commencement of processing. The priority of the actions themselves remain to be established dynamically according to the data. This dynamic strategy element can have one of two states, namely, merging or splitting, depending on which action will have the higher priority setting.

2.1.5.2. Line Rule Priority

The argument described above for the REGION rules also applies here. A dynamic ordering is also imposed by the actions of the rules, and a static ordering is imposed by the conditions of rules with the same type of action. The types of actions used are, of course, different. Again, two types are distinguishable, those that add new

---

*Note that a line that is near another must be in front, behind, or parallel to it.

lines, and those that delete existing ones. The former includes actions that insert, extend, and join lines, while the latter includes those that delete a line and merge two lines into one. The two states of this element of dynamic strategy will thus be referred to as the add and delete states.

This concludes our discussion of the various elements of the local strategy of the system. These elements can be dynamically modified to reflect any required combination of states [Levine and Nazif, 1982 b]. Figure (1) summarizes the strategy elements and their possible states.

Region Path Strategy	highest adjacency largest adjacent smallest adjacent raster scan higher label
Line Path Strategy	closest in front closest behind closest parallel longest near shortest near strongest near weakest near raster scan higher label
Sequential/Parallel Strategy	sequential parallel
Processor Priority	regions lines
Region Rule Priority	merge split
Line Rule Priority	add delete

Figure (1) : The dynamic elements of local strategy and their states

3. STRATEGY DETERMINATION

Measuring performance is a key factor in providing a feedback path by which a system can modify its strategy during processing. Elsewhere, we introduce a set of segmentation measures which serve as performance parameters for the system [Levine and Nazif, 1982 b]. These measures collectively take into account uniformity within regions, contrast across regions, as well as making provision for lines and texture. They have the additional property of being dynamic, in the sense that they reflect the state of the segmentation at any point in time. When evaluated over each area of attention in the image, they constitute a performance vector, whose utility in determining local and global strategies will become apparent.

The performance vector for each area in the image will include components for region uniformity, region contrast, line contrast, line connectivity, number of regions, and number of lines. Note that the first three components are computed per feature in the image. For black and white images, each component can be computed for the grey level intensity of regions and lines. For color images, each component will contribute three elements, to the performance vector, one for each color channel intensity. In general, the uniformity and contrast of other features of regions and lines can be added to the performance vector. For m such features, the performance vector $\bar{P}_a$ for area a will have M = 3m+3 components, and is given by

$$\bar{P}_a = [U_a^1,\ldots,U_a^m,C_a^1,\ldots,C_a^m,H_a^1,\ldots,H_a^m,T_a, R_a,L_a], \quad (1)$$

where U_a^i, C_a^i, and H_a^i are region uniformity, region contrast, and line contrast measures for feature i in area a, T_a is the line connectivity measure for a, R_a and L_a are the number of regions and lines in a, respectively. Each area a will be represented in the strategy determination process by its performance vector $\bar{P}_a$ and its type ψ_a*.

The strategy of the rule based system is determined in two steps. First, a global strategy evaluation scheme selects the next area of attention in the image. Once this is done, the local strategy used by the system within the selected area must be computed. This section details the methodology for determining the system strategy at both levels. A solution for global strategy determination is presented in the form of a fuzzy decision-making process. For a detailed introduction to the fundamentals of fuzzy logic, the reader is referred to [Zadeh, 1973; Zadeh, 1976; Gaines, 1976], and the recent bibliography on the subject and related applications given by [Gaines and Kohout, 1977].

The constraints that govern the choice of a particular local strategy, and define the global strategy for area selection, will be described. The fact that these constraints are themselves fuzzy in nature, will be shown to be the reason for using fuzzy logic, rather than other standard decision-making methodologies. The problem of evaluating the local strategy is posed in the form of a "black box" model with specific inputs, outputs, and guiding constraints. The same principles used for global strategy selection are also used to solve for the local strategy model.

3.1 Global Strategy Evaluation

Before determining the local strategy within an area in the image, the area must be selected. A global process should examine all available areas in the image, and select, based on their properties, the most appropriate one. One global strategy is to select the area that: (i) is in more need of further processing, and (ii) has been less recently visited by the system. An area's need for processing is reflected by its performance parameters. The first four parameters described in equation (1) serve this purpose fully. The amount of additional region processing that an area requires is reflected in how low the region uniformity and contrast measures are. Low values of line connectivity and contrast measures

---

* Areas are classified as smooth, textured, or as bounded by closed lines, see [Levine and Nazif, 1982 a].

indicate the need for adding or deleting more lines, respectively. The system should thus select the area with the lowest overall values for these four parameters.

A parameter that reflects temporal order, rather than performance, is required in order to represent the second constraint mentioned above. Consider the recency ratio for area a given by

$$R_a = \frac{\text{Number of areas visited since a was}}{\text{Total number of areas in the image}} \quad (2)$$

A high value of R_a would indicate that the area has not been visited recently, and thus is an eligible candidate for processing.

Our global strategy, put in words, is to select the area of lower region uniformity and contrast, lower line connectivity and contrast, and of higher recency ratio. Although we can express each of these measures mathematically, the constraint over each of them is not so well-defined. The format is a fuzzy one, since it only indicates that a measure should be low or high. Furthermore, the method by which the measures should be combined to have an effect on the final decision, yields another source of imprecision. We know that an area should satisfy as many of the constraints as possible, in the best possible way. In practical situations, each area will satisfy each of the constraints to some degree. Thus, a precise mathematical function of the measures could be formulated to produce a combined effect. However, such a formula, involving multiplicative or additive effects, would be too simplistic and too precise. This is because we only have a fuzzy notion of what to expect. Too fuzzy, in fact, to be represented by a single-valued mathematical function. The choice is thus better described by using a linguistic framework, as opposed to a multi-variable formula. This provides the motivation for using the concepts of fuzzy set theory in making the decision.

The performance parameters of equation (1) were designed so that every measure would indicate how well an area satisfies a particular objective. A region uniformity measure of 0.8 for an area, for example, indicates that we are 80% confident that the regions in the area are uniform. The same applies to the other region and line measures (note that they are all normalized to lie in the interval [0,1]). The confidences required by the fuzzy decision-maker can be derived directly from these measures. If the value of a performance measure for an area is x, the amount of further processing required to improve that measure is proportional to 1 - x. The recency ratio defined by equation (2) provides a confidence value that is proportional to an area's temporal eligibility.

For each area, M = 3m + 2 objectives are defined, where m is the number of features of regions and lines used for the evaluation, as introduced by equation (1). They correspond to all m region uniformity, m region contrast, m line contrast, one line connectivity, and one recency ratio measures. Let A be the set of N areas in the image:

$$A = [\alpha_1, \alpha_2, \ldots, \alpha_N], \quad (3)$$

and $Y_1, Y_2, \ldots, Y_M$ be the set of M objectives. Following the notation in [Zadeh, 1976], we have for each α_i,

$$Y_k = \int_A (1 - U^k_{\alpha_i})/\alpha_i, \quad 0 < k \leq m, \quad (4\text{-a})$$

$$Y_k = \int_A (1 - C^{k-m}_{\alpha_i})/\alpha_i, \quad m < k \leq 2m, \quad (4\text{-b})$$

$$Y_k = \int_A (1 - H^{k-2m}_{\alpha_i})/\alpha_i, \quad 2m < k \leq 3m, \quad (4\text{-c})$$

$$Y_k = \int_A (1 - T_{\alpha_i})/\alpha_i, \quad k = 3m + 1, \quad (4\text{-d})$$

$$Y_k = \int_A R_{\alpha_i} / \alpha_i, \quad k = 3m + 2, \quad (4\text{-e})$$

where U^k_α, C^k_α, H^k_α, and T_α, are defined as per equation (1). The function inside the integral indicates the degree of membership of α_i in the fuzzy set representing objective Y_k.

The decision-making process takes place by first computing the degree by which all areas satisfy each of the M objectives. For a particular objective k, this will correspond to finding the fuzzy subset Y_k over the set of areas A:

$$Y_k = \left[\frac{y_{1k}}{\alpha_1}, \frac{y_{2k}}{\alpha_2}, \ldots\ldots, \frac{y_{Nk}}{\alpha_N} \right] \quad (5)$$

The y_{ik} are computed using equations (4-a) to (4-e), depending on the value of k. The fuzzy decision set D is then computed from the M sets of objectives, so that

$$D = Y_1 \cap Y_2 \cap \ldots\ldots \cap Y_M, \quad (6)$$

$$\text{or} \quad D = \left[\frac{d_1}{\alpha_1}, \frac{d_2}{\alpha_2}, \ldots\ldots, \frac{d_N}{\alpha_N} \right], \quad (7)$$

where

$$d_i = \underset{k=1}{\overset{M}{\text{MIN}}} [y_{ik}]. \quad (8)$$

Following the optimal decision rule, the system will select area α_j as the next area to process, if

$$d_j = \underset{i=1}{\overset{N}{\text{MAX}}} [d_i]. \quad (9)$$

The maxi-min process of fuzzy decision-making is given by equations (8) and (9). The objective that is minimally satisfied by an area is selected to represent it. The area with the highest value for the latter will constitute the output of the decision-maker.

3.2 Local Strategy Evaluation

Once the global strategy has selected an area in the image, it remains to define the local strategy within the area. In section 2, we described the elements of local strategy, their states, and how they can be adjusted by the system. We have also introduced in this section, a performance vector that can be evaluated for different areas in the image. The components of this vector have been designed to measure the quality of a segmentation over an area, and thus provide the means for adjusting the local strategy within that area. They can do so by influencing the individual strategy elements into particular states that are more suitable for the data configuration in the area, as represented by these parameters. The following problem presents itself: Given a performamce vector $\bar{P}_a$ for area a, how can we determine the elements of the strategy vector $\bar{S}_a$ for that area. What we are actually looking for is the model F represented by the "black box" in figure (2).

Figure (2) : Model for strategy evaluation

The constraints imposed on this model are brought about by the manner in which each performance parameter affects each strategy element. One can say that line analysis should have priority over region analysis if the region measures of uniformity and contrast are high, and the line measures of contrast and connectivity are low. This is an indication that the image needs more line processing than region processing. The opposite is also true, and variations on the above are possible. More splitting is needed at low region uniformity and more merging at low region contrast. Adding lines is preferable for low connectivity areas, while deleting lines is more useful in smooth areas. A sequential strategy is more efficient if the amount of processing left is low, as reflected by high values for the region and line measures. On the other hand, the more conservative parallel strategy is safer if the measures are low, and if the number of regions and lines are high. Bounded areas may invite region merging, while textured areas allow for more line deletion. The path strategy is also affected by these constraints. While low region uniformity suggests visiting larger regions first, low region contrast suggests moving through high adjacency values. Smaller size regions are more influential for texture areas, while the opposite is true for smooth areas.

These are just examples of the types of constraints present, and are not meant to constitute an exhaustive list. Clearly, not every performance parameter affects every strategy element, at least not to the same extent. Some constraints seem more important than others. Two factors affect the presence and relative importance of these constraints: improving the efficiency of the computation, and enhancing the quality of the output segmentation.

A consistent relationship can be obtained by generalizing the effects of performance parameters on strategy elements. This can be done by explicitly naming the effect of every parameter on all the states of all strategy elements. Two aspects can be coded: the direction of the effect, and its magnitude. A performance parameter can promote the instantiation of a particular state for one of the strategy elements in one of two different ways. The higher the value of the parameter, the more it would advocate the choice of that state. This will be termed a positive effect. The second and opposite effect, when a lower parameter value is more suggestive of the selection of the affected state, is termed the negative effect.

Table (1) represents the direction of the relation between performance parameters and strategy elements. The "+" and "-" signs correspond to the positive and negative effects, respectively. An entry is included whenever the direction of influence of a parameter on a state is known, irrespective of the magnitude of that influence. If no known effect exists, however, the entry in the table is left blank.

As indicated previously, the extent by which performance parameters affect strategy elements is not constant. Some parameters are more important than others for certain elements.

Strategy	Performance Parameters					
Elements	U	C	H	T	R	L
Adjacency	+	-			-	
Largest	-	-			+	
Smallest	+	+			-	
By Label	+	+			+	
Nearest			+	-		+
Longest			-	+		-
Shortest			+	-		+
Strongest			+	-		-
Weakest			-	+		+
By Label			+	+		+
Sequential	+	+	+	+	-	-
Parallel	-	-	-	-	+	+
Regions	-	-	+	+	+	-
Lines	+	+	-	-	-	+
Merge	+	-	-	-	+	-
Split	-	+	+	+	-	+
Add	-	+	+	-	+	-
Delete	+	-	-	+	-	+

Table (1) : Effect of performance parameters on the states of strategy elements.

The relative influence of the parameters on the elements will also depend on the type of area for which the strategy is evaluated. Table (2) summarizes this magnitude effect. The entries in the table code the influences into one of five categories: very low (VL), low (L), medium (M), high (H), and very high (VH). Variation due to area type is represented by including three values (one for each type of area), to indicate the magnitude of the effect a performance parameter has on each of the six elements of strategy.

Strategy Elements	Area Type	Performance Parameters U	C	M	T	R	L
Region	smooth	M	M	VL	VL	M	VL
Path	texture	L	M	VL	VL	VH	VL
Strategy	bounded	M	L	VL	VL	VH	VL
Line	smooth	VL	VL	L	M	VL	M
Path	texture	VL	VL	M	M	VL	VH
Strategy	bounded	VL	VL	M	M	VL	M
Parallel/	smooth	M	M	L	M	M	M
Sequential	texture	L	M	M	M	M	M
	bounded	M	L	M	M	M	M
Process	smooth	VH	M	M	M	VL	VL
Priority	texture	M	VH	M	VH	L	L
	bounded	M	M	M	M	L	VL
Region	smooth	VH	M	L	L	M	VL
Rules	texture	M	VH	L	L	VH	VL
Priority	bounded	M	M	L	L	VH	VL
Line	smooth	L	L	VH	M	VL	M
Rule	texture	L	L	M	VH	VL	VH
Priority	bounded	L	L	M	M	VL	M

Table (2) : Codes that reflect the relative importance of performance parameters in determining the various strategy states in different areas

The concepts of fuzzy logic will now also be extended to the problem of local strategy determination. Unlike the global strategy, more than one decision is required. Six elements must be instantiated in order to completely specify the local strategy. Therefore, as many decisions are needed to assign a state to each of these elements. The model F is thus composed of six independent, yet identical, decision processes, that can be executed in parallel. Each process involves the selection of one of a finite number of states, and hence the determination of one component of the local strategy vector. The following analysis will apply equally to each of the strategy elements.

Let the set of n states for strategy element S_i be given by

$$S_i = [\, s_{i1}, s_{i2}, \ldots, s_{in} \,]. \qquad (10)$$

We can construct M fuzzy subsets of S_i. Each subset corresponds to one of M = 2m+3 parameters constituting an area's performance vector. Each subset will have the form

$$Y_k = [\, \frac{\mu_{k1}}{s_{i1}}, \frac{\mu_{k2}}{s_{i2}}, \ldots, \frac{\mu_{kn}}{s_{in}} \,], \quad 1 \leq k \leq M \qquad (11)$$

where μ_{kj} is the degree by which performance parameter k satisfies the requirements of state j for strategy element i in the given area. Each parameter is evaluated as a numerical quantity in the interval [0,1]. If a parameter has a positive effect on a state, then the higher the value of that parameter, the more the degree of satisfaction of that state with it. For a negative effect, that degree would increase with lower parameter values. If parameter k is found to have a value P_k, then

$$\mu_{kj} = \begin{cases} P_k & \text{if k has a positive effect on j,} \\ (1-P_k) & \text{if k has a negative effect on j,} \end{cases} \qquad (12)$$

as specified by the entries of Table (1).

The decision objectives represented by the M fuzzy subsets of equation (11) can now be computed from the performance parameters of the area. They can then be used to compute a fuzzy decision set D_i on the states of strategy element i, so that

$$D_i = Y_1^{\gamma_{i1}} \cap Y_2^{\gamma_{i2}} \cap \ldots \cap Y_M^{\gamma_{iM}}, \qquad (13)$$

where γ_{ik} is a quantity that reflects the relative importance of performance parameter k, in determining strategy element i, according to Table (2). A positive power that is greater than unity, will emphasize the influence of a particular objective. A fractional power will have the opposite effect.

The last four strategy elements in Table (1) are binary valued. The entries that are positively supporting one state, are seen to have a negative effect on the other state for these elements. The first two elements are multi-valued, and each parameter can support more than one state, in a positive or negative manner, simultaneously. It is the integrated effect over all parameters that tips the balance towards a particular state. This is, of course, a mutually exclusive process, since a strategy element can only assume a single state at a time. The fuzzy decision set D_i computed in equation (13), provides the degree of satisfaction of the decision-maker with each of the states of strategy element i. The state with the maximum satisfaction value is chosen to represent that element in the output strategy vector. This process is repeated for all six strategy elements.

The model F in figure (2) can thus be represented by a set of modules, each of which computes the degree of satisfaction of each strategy element state, with the current value of each performance parameter. These are followed by a maxi-min selection module for every element.

4. CONCLUSIONS

In the previous sections, we have discussed the various factors involved in setting the strategy of our rule based image processing system. We have demonstrated the difficulties involved in getting such a complex yet versatile system to function in an optimized fashion. To do this, we have analyzed the system strategy into basic, well defined, and independent elements, that can acquire a number of finite and specific states. Next we defined a set of parameters that measure the performance of the system, as reflected by the state of the output at any point during processing. A model is then introduced that uses these parameters in evaluating decision functions that allocate states to the various strategy elements, thus defining a specific strategy for the system.

The solution proposed involves a novel approach to both rule-ordering in a rule based production system, and data ordering in image processing systems. The system is designed to allow for dynamic setting of its rule matching priorities and its focus of attention scheme. A pool of focus of attention areas are generated according to certain criteria embedded in rules that are designed for this purpose. Performance parameters are subsequently computed for each area, based on the initial segmentation status within that area. These parameters are stored and dynamically updated during processing, so that they can continuously and appropriately represent each area at any time. Once an area is selected, the system proceeds to define the local strategy within that area. It evaluates decision functions that reflect the effect of the performance parameters of the area, on each of the individual strategy elements. Both local and global evaluations employ a fuzzy set decision-making methodology to account for the inherent imprecision in the constraints representing the relation between the data and the resulting decisions.

The key factor in creating a dynamic strategy is the ability to dynamically measure the state of the segmentation, and consequently obtain an indication of the distance from the final goal [Levine and Nazif, 1982 b]. By implementing such a strategy in our rule based image processing system, we are serving a dual purpose. First we ensure efficient processing by modifying the areas in an appropriate order. The second goal is to minimize the segmentation errors by selecting the best rules at the proper point in the computation.

REFERENCES

Gaines, B.R., Foundations of Fuzzy Reasoning, Int. J. Man-Machine Studies, Vol. 8, 1976, pp. 623-668.

Gaines, B.R., Kohout L.J., The Fuzzy Decade: A Bibliography of Fuzzy Systems and Closely Related Topics, Int. J. Man-Machine Studies, Vol. 9, 1977, pp. 1-68.

Levine, M.D., Nazif, A., An Experimental Rule Based System for Testing Low Level Segmentation Strategies, in "Multi-Computer Architectures and Image Processing: Algorithms and Programs", L. Uhr and K. Preston, (eds.), Academic Press, N.Y., 1982 a.

Levine, M.D., Nazif, A., Performance Measurement and Strategy Evaluation for Rule Based Image Segmentation, TR-82-1, Computer Vision and Graphics Lab., Department of Electrical Eng., McGill University, March 1982 b.

Nazif, A., Levine, M.D., A Rule Based Low Level Segmentation System, Seventh Canadian Man-Computer Communication Society Conference, June 10-12, 1981, Waterloo, Ontario, Canada.

Zadeh, L.A., Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, Jan. 1973, pp. 28-44.

Zadeh, L.A., A Fuzzy-Algorithmic Approach to the Definition of Complex or Imprecise Concepts, Int. J. of Man-Machine Studies, Vol. 8, 1976, pp. 247-291.

Syntactic Pattern Analysis As A Means of Scene Matching

John F. Gilmore

Martin Marietta Orlando Aerospace

Abstract

This paper describes a technique for matching two images of natural terrain using a syntactic pattern recognition approach. Points of interest in an image are classified and a graph possessing properties of invariance is created based on these points. A method for generating a grammar string from the classified graph structure is presented Local match analysis is performed and the best global match is constructed. A probability-of-match metric is computed in order to evaluate the global match and results demonstrating these steps are presented.

Introduction

In order to match two images of natural terrain, the differences between images must be explored. The images to be matched may differ in viewing angle, lighting and weather conditions. Based upon the application, the images may also have been obtained with different sensors. As a result of these variations, matching image by correlating the pixel intensity values is not effective in the presence of such large differences as the partial obscurations that occur in natural terrain because of varying perspective angles. An alternate approach must be explored to solve this problem.

This paper deals with the application of syntactic pattern recognition as it pertains to scene matching. The goal of this work is to devise a method for matching a sensed image with a reference image. Because the sensed image may not be contained in the reference image, a probability of match must be determined and a threshold selected so that each sensed image can be evaluated independently.

The syntactic approach to pattern recognition has attracted increasing interest in recent years. Several texts have been published which present syntactic recognition fundamentals (1-3). Many articles dealing with applications to specific areas have also been presented. Tai and Fu (4) investigated class inference in relation to context-free programmed grammars for syntactic pattern recognition. Tsai (5) combined statistics with a syntactic approach to recognize industrial objects. Mohr and Masini (6) investigated syntactic recognition as a strategy for scene analysis using knowledge directed recognition. Other methods have also been explored (7-10).

Syntactic Pattern Analysis System

We can best understand the syntactic method proposed here by comparing it with the standard pattern recognition approach, which consists of four basic operations. First, an acquired image is input to a sensor. Second, some type of image preprocessing such as segmentation or image enhancement is applied. Third, a set of predetermined statistical features are extracted from the image. Fourth, regions of the image are classified based on their statistical properties. This method is illustrated in Figure 1.

Fig. 1 Standard Pattern Recognition Approach

The syntactic approach to the pattern recognition problem consists of six distinct sections:

1. Point classification
2. Graph generation
3. Grammar string formation
4. Match analysis
5. String comparison
6. Probability of match

These sections have been combined to form the Syntactic Pattern Analysis (SPA) system. The SPA approach (Figure 2) combines spatial relationships and scene matching.

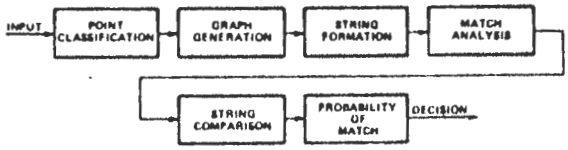


Fig. 2 Proposed Syntactic Recognition Approach

1. Point Classification

In order to work with point patterns, the points of interest in an image must be identified and classified into distinctive classes. The classification process is two-fold. First, the

image is preprocessed to identify and segment areas that are deemed classifiable. Once these regions have been segmented, a set of features is determined for each. Based upon how well each feature set compares with a predetermined set of feature vectors, a region is classified as a tree, field, haystack, river, lake, man-made object, moving object, or clutter. Features used in this classification include perimeter, area, height, width, Euler numbers, texture, contrast measures, and intensity moments. Once identified, a representative location point such as the centroid is chosen for each region.

The second phase of the classification process deals with those regions which have been classified as clutter. Because the majority of images encountered contain a relatively small number of nonclutter points, an effort is made to identify three subclasses of clutter. A subset of the original feature values from each region is submitted to a K-means classifier. The K-means algorithm is based on the minimization of a performance index, in this case the sum of the squared distances between all points in a cluster domain and the cluster center. With K set to a value of three, the algorithm will iteratively produce three distinct subclasses of clutter. When all of the subclasses have been identified, the centroid of each region is chosen as the representative point location for that region.

2. Graph Generation

After all of the points of interest are classified, they are analyzed in terms of graph generation. The application requires symbolic graphs possessing properties invariant to translation, rotation, and magnification. Large changes in perspective have a dramatic effect on the symbolic graph and therefore must also be addressed. A nearest-mean algorithm has been developed to handle perspective changes in two dimensions. Further development of this method shows promise in dealing with the three-dimensional world.

Graph generation is a two-phase process involving both the internal and external graph structure. An internal graph representative such as the two-nearest-neighbors is applied to the classified points of interest. The external hull is chosen when all of the points have been processed. The boundary hull or the convex hull are the only two choices available. The boundary hull traces the perimeter of the internal structure, recording each node classification as it progresses. The hull is completely determined when the initial starting point is re-encountered.

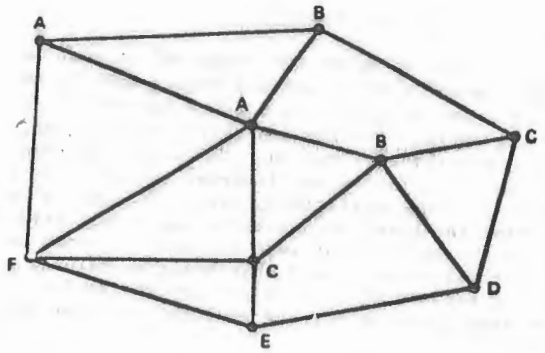
The convex hull algorithm tests a series of three points for convexity. If a point is deemed to be part of the hull, it is recorded and a path from the last convex point is constructed (11). By combining the internal and external graphs of a point pattern, a symbolic representation of the original image is created.

3. Grammar Generation

Syntactic processing dictates that the symbolic graphs generated from the point patterns be transformed into a grammar string formation. One method of accomplishing this is to select a node and make a clockwise sweep of all possible circuits that flow through the given node. Each circuit would be recorded individually as a string of point classifications. This process is repeated for all of the nodes in the graph based on three priorities:

- o First, the hull nodes are recorded as a circuit.
- o Second, all subgraphs with at least one hull node are analyzed.
- o Third, all subgraphs with no hull nodes are identified.

An example of the grammar extraction process is shown in Figure 3. The string grammars formed by this method constitute regular grammars and, as such, possess rotational image matching capabilities.



I ABCDEF
II ABA, CDB, EFC, FAA, FAC
III ABC

Figure 3. Prioritized Grammar Extraction

4. Match Analysis

A one-to-one subgrammar comparison of the sensed-grammar string to the reference-grammar string is performed in the match analysis. Productions of the regular grammar are possible due to point misclassification but are recorded as transformations instead because of the unique application of the grammar. These transformations are hierarchical in nature; that is, a moving object may be classified as a man-made object but a man-made object will never transform to a moving object. Particular attention is paid to the number of false negatives and false positives encountered by each local match

N = 25 FALSE POSITIVES = 6 NUMBER OF SAME = 24

M = 30 FALSE NEGATIVES = 1 TRANSFORMATIONS = 0

$$\text{PROBABILITY} = \left(\frac{N+M-FP-FN-0.5T}{N+M}\right)\left(\frac{NS}{N}\right) \cdot \left(\frac{25+30-6-1-0}{25+30}\right) \cdot \left(\frac{24}{25}\right) = 0.837$$

Figure 5. String Comparison Probability Computation

because they will noticeably affect match performance. Figure 4 demonstrates a scoring criteria when a false positive is placed in the sensed image. In this case, S4 would match R4 and S5 would be disregarded.

REFERENCE	SENSED	MATCH SCORES
1) A B C A C	1) A B C X A C	S1 R1 0.909
2) A B E D	2) A B E D	S2 R2 1.000
3) B C E	3) B C E	S3 R3 1.000
4) C A E	4) C X E	S4 R4 0.444
5) A C D	5) X A D E	S5 R4 0.381
6) C A D D	6) A C D	S6 R5 1.000
7) D D E	7) C A D D	S7 R6 1.000
	8) D D E	S8 R7 1.000

Figure 4. Match Analysis Scoring Criteria

5. String Comparison

A global match analysis is performed based on the local probability of match. The local probabilities are computed for each substring in the match analysis section to identify the best sensed-to-reference substring match. Only those probabilities which exceed a realistic threshold are maintained. This approach is taken in order to analytically determine the best overall string match, rather than take the first substring match found, and is an excellent means of dealing with the false negatives and false positives encountered. These false values would otherwise deteriorate the global match as it progressed through the substrings. The individual substring probabilities are evaluated and the best overall match determined. When the global match is complete, the total number of false negatives, false positives, and transformations are determined.

6. Probability of Match

The probability of match for the sensed image as it relates to the referenced image is calculated based upon the following non-normalized equation:

N = the number of elements in referenced image

M = the number of elements in sensed image

NS = number of elements in the reference image that were identically matched in the sensed image

FN = the number of false negatives encountered

FP = the number of false positives encountered

T = the number of element transformations encountered.

$$\text{Probability of Match} = \frac{(N+M-FN-FP-.5T)}{(N+M)} * \frac{(NS)}{(N)}$$

The probability of match value is compared to an experimentally determined threshold and a decision made whether an exceptable match has been found. In the case of matching multiple sensed images to a referenced image, the image with the highest probability would be the best match, based on the aforementioned criteria. An example of a string comparison and its probability is shown in Figure 5.

Classification Experiment

A classification experiment to demonstrate the matching capabilities of the SPA system using FLIR imagery was performed. A reference

image was chosen (Figure 6) and a subimage corresponding to a sensed image (Figure 7) was extracted. The reference image was preprocessed and the following points of interest and associated classes were generated:

Point Location	Class	Classification
(203, 14)	A	clutter
(176, 16)	A	clutter
(186, 62)	C	clutter
(107, 80)	C	clutter
(226, 84)	A	clutter
(247, 89)	A	clutter
(154, 99)	B	clutter
(154, 111)	B	clutter
(27, 164)	A	clutter
(19, 218)	B	clutter
(23, 255)	C	clutter
(241, 281)	C	clutter
(210, 308)	B	clutter
(182, 217)	T	target
(265, 170)	H	haystack
(154, 194)	H	haystack

This process was repeated for the sensed image yielding the following point locations and classifications:

Point Location	Class	Classification
(17, 59)	A	clutter
(13, 150)	C	clutter
(231, 126)	C	clutter
(9, 63)	B	clutter
(200, 173)	B	clutter
(172, 62)	T	target
(144, 39)	H	haystack
(255, 15)	H	haystack

For each classified point pattern, a convex hull and a two-nearest-neighbor graph were generated. Figures 8 and 9 are the symbolic reference and sensed graphs, respectively. Grammar strings were generated for both graphs and the resulting match is indicated below.

Reference Graph	Sensed Graph	Match Scores	
1) ABCBCHAAAC	1) ABCBCH	S1-R1	0.45
2) ABCTH	2) ABCTH	S2-R2	1.00
3) CBT	3) CBT	S3-R3	1.00
4) BCT	4) BCT	S4-R4	1.00
5) CHT	5) CHT	S5-S5	1.00
6) HHT	6) HAH	S6-R6	0.44
7) ACA			
8) AAC			
9) AAC			
10) CBB			

Probability of Match = 0.72

Summary

The syntactic pattern recognition problem has always been viewed as impractial in application, based on the results achievable using a atatistical approach. In the field of scene matching, this approach has led to systems which are incapable of dealing with large changes in perspective or noise-induced false negatives and false positives. Additionally, no true means of dealing with statistical misclassification have been developed. This has resulted in the occurrence of a high number of false alarms. The Syntactic Pattern Analysis system addresses these problems in an analytical manner and shows exceptional promise for further work.

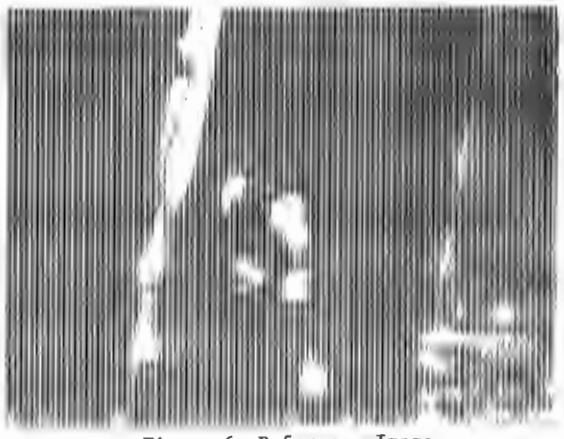


Figure 6. Reference Image

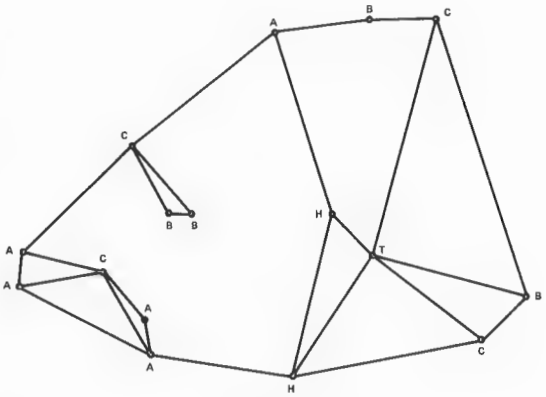


Figure 8. Symbolic Reference Graph

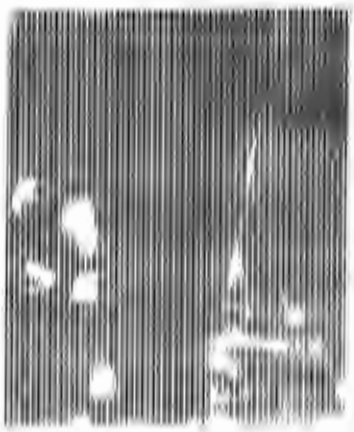


Figure 7. Sensed Image

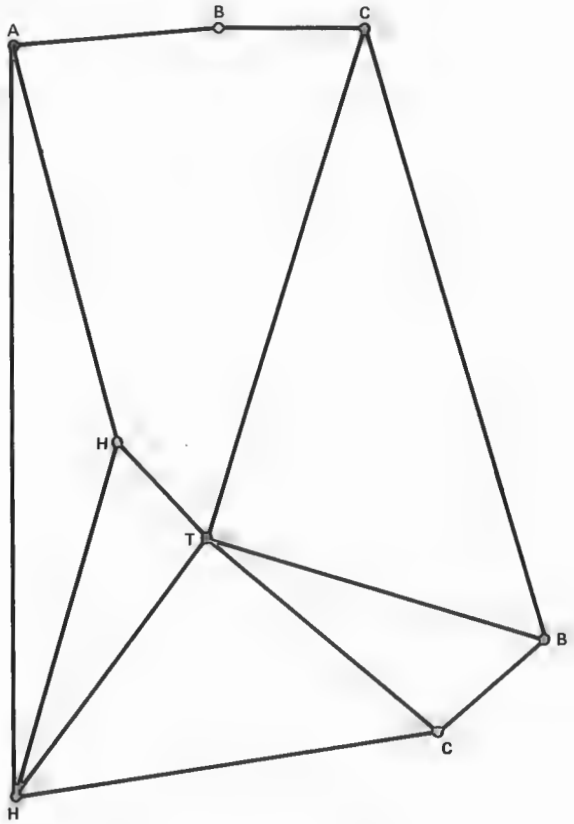


Figure 9. Symbolic Sensed Graph

Bibliography

(1) Gonzalez, R. C. and Thomason, M. G.
Syntactic Pattern Recognition
Addison-Wesley, Massachusetts, 1978

(2) Fu, K. S.
Syntactic Methods in Pattern Recognition
Academic Press, New York, 1974

(3) Pavilidis, T.
Structural Pattern Recognition
Spring-Verlag, New York, 1977

(4) Tai, J. W.
"Inference of a Class of Context-Free Programmed Grammar of Syntactic Pattern Recognition"
Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, p. 18
Dallas, Texas, August 1981

(5) Tsai, W. H. and Fu, K. S.
"A Syntactic-Statistical Approach to Recognition of Industrial Objects"
Proceedings of the Fifth ICPR, p. 251
Miami, Florida, December 1981

(6) Mohr, R. and Masini, G.
Knowledge Directed Recognition: A Syntactical Approach"
Proceedings of the Fifth ICPR, p. 337
Miami, Florida, December 1981

(7) Bhargava, B. K. and Fu, K. S.
"Tree Systems for Syntactic Pattern Recognition"
IEEE Transactions on Computers
Vol. c-22, p. 1087, December 1973

(8) Moayer, B. and Fu, K. S.
"A Syntactic Approach to Fingerprints Pattern Recognition"
Pattern Recognition
Vol. 7, p. 1, 1975

(9) You, K. C. and Fu, K. S.
"A Syntactic Approach to Shape Recognition Using Attributed Grammars"
IEEE Transactions on Systems, Man and Cybernetics
Vol. SMC-9, No. 6, June 1979

(10) Pavilidis, T.
"The Use of a Syntactic Shape Analyzer for Contour Matching"
IEEE Transactions on Pattern Recognition and Image Processing
Vol. PAMI-1, No. 3, p. 307, July 1979

(11) Akl, Selim G. and Toussaint, Godfriend T.
"Efficient Convex Hull Algorithms for Pattern Recognition Applications"
Proceedings of the Fifth ICPR, p. 126
Miami, Florida, December 1981

An Error Correcting Contextual Algorithm for Text Recognition

RAJJAN SHINGHAL

Department of Computer Science, Concordia University
1455 De Maisonneuve Blvd. West Montreal, Quebec, Canada H3G 1M8

ABSTRACT

A text-recognition algorithm is described which uses a dictionary to correct the substitution errors of a Markov method. Experiments were conducted with the algorithm on texts of both conventional and unconventional underlying statistical structure. The letters comprising the texts were of varying quality: well-written and badly written. The performance of the algorithm is compared to its complexity.

I. INTRODUCTION

A common error in OCR (Optical Character Recognition) application is that of substitution: a letter being recognized as some other letter. In text-recognition, substitution error is extremely undesirable because it causes a spelling error in the word being recognized, and even a low letter error-rate can lead to a high word error-rate [5]. Quite a few researchers [2], [3], [5], [7], [9], [10], [11], [17], [18] have identified different kinds of spelling errors and presented techniques to correct such errors. For OCR application Hanson et al. [5] used positional binary N-grams in a contextual postprocessor and succeeded in reducing the word error rate substantially for a data base of 6-letter words. However, the memory required to store the binary N-grams is fairly large because of the combinatorial problem caused by the various combinations of positions for which the binary N-gram are required.

This paper presents an algorithm to correct some substitution errors in OCR application of text recognition. It is a hybrid algorithm as it combines the modified Vitebi algorithm [13] with a dictionary recognition method [1], [4]. Thus it requires that the words to be recognized exist in a previously compiled dictionary. The hybrid algorithm is described in Section II. The algorithm was experimentally tested on English texts of different statistical structures. The passages of text were assembled by using handprinted letters of varying quality. The experiments are described in Section III. The conclusion from the experiments are given in Section IV.

II. THE PROBLEM OF TEXT RECOGNITION

The text to be recognized consists of the patterns of the characters. A set of measurements (called the feature vector) is made on each pattern

Let $\bar{X} = X_0, X_1, X_2, \ldots, X_n, X_{n+1}$ be a sequence of such feature-vectors ($n \geq 1$) which are presented sequentially to the recognizer. Let X_0 and X_{n+1} be associated with the character '#', and X_1 to X_n with letters A to Z. X_1 to X_n thus form a word in the text. Let $P(\bar{X}|\bar{\lambda}=Z)$ denote the probability of $\bar{X}$ conditioned on the sequence of pattern classes $\bar{\lambda} = \lambda_1, \lambda_2, \ldots, \lambda_n, \lambda_{n+1}$ taking on the values $\bar{Z} = Z_0, Z_1, Z_2, \ldots, Z_n, Z_{n+1}$. For simplicity in notation, $P(\bar{X}|\bar{\lambda}=\bar{Z})$ shall be written as $P(\bar{X}|\bar{Z})$. The probability of recognizing $\bar{X}$ correctly is maximized by selecting that sequence of characters which maximizes the a posteriori probability $P(\bar{Z}|\bar{X})$ or a monotonic function of it, say $\log P(\bar{Z}|\bar{X})$. By assuming that blanks are perfectly recognizable (that is, $Z_0 = Z_{n+1} = $ '#'), that the feature-vectors are conditionally independent, and that letters in a word form a Markov chain of order 1, it can be shown [13] that maximizing $\log P(\bar{Z}|\bar{X})$ is equivalent to maximizing $g(\bar{X},\bar{Z})$, where

$$g(\bar{X},\bar{Z}) = \sum_{i=1}^{n} \log P(X_i|Z_i) + \sum_{i=1}^{n+1} \log P(Z_i|Z_{i-1}).$$

In the equation above, $P(X_i|Z_i)$ is called the likelihood, and $P(Z_i|Z_{i-1})$ is called the transition probability (the probability of Z_i occurring to the immediate right of Z_{i-1} in text). Let $g(\bar{X},\bar{Z})$ be called the score of the word $\bar{Z}$. Computing the scores of the 26^n possible words, requires $2n \times 26^n$ additions, which is about 48.3 million additions for an average word length of 4.74 [14]. This is considered to be too high a complexity. One way to reduce this complexity is to maximize $g(\bar{X},\bar{Z})$ approximately; that is maximize $g(\bar{X},\bar{Z})$ over only a selected subset of the 26^n possible words. Different text recognition algorithms may use different criteria in selecting this subset.

Four such algorithms were compared on their performance and complexity [12]. These four algorithms were Noncontextual, Heuristic approximation, Modified Viterbi, and Uniform Pruning. It was concluded that for its complexity, the Modified Viterbi Algorithms (MVA) gave the best performance. Thus for the experiments in this

paper, it was decided to combine the MVA with a Dictionary Algorithm (DA), see [1], [2], [4], [18].

The MVA has been completely described in [3]. So it is not being described here. Informally, it consists of $d \geq 1$ alternatives (d is a heuristic) being selected as candidates for each X_i in the n-letter word. A path is then traced through the d by n trellis such that at each X_i, only the d most likely letter strings ending with the d alternatives of X_i are retained. Thus unlikely letter strings are pruned from consideration. In the DA based on [1], $g(\bar{X},\bar{Z})$ is maximized over the n-letter words in the dictionary. The MVA and DA were combined in [14] too, but here a further change was made. It was observed that the output from the MVA suffered from errors of substitution. So the strings output by the MVA are searched for in the dictionary to find the best match. Thus it corrects errors. The algorithm in now described below:

Algorithm E(Error Correcting Hybrid Algorithm)

E1. Recognize the input pattern sequence $\bar{X}$ by the MVA. The MVA outputs d ($d \geq 1$) n-letter strings $S_1, S_2, \ldots, S_d$ such that $S_i (1 \leq i \leq d)$ is the i^{-th} most likely string into which $\bar{X}$ can be recognized. {Comment: The value of d is a heuristic and is predecided by the user. Some typical values of d range from 2 to 5, see [13].}

E2. Perform Steps E3 through E4 for $i=1,2,\ldots,d$.

E3. Search S_i in dictionary. If search is successful recognize $\bar{X}$ as S_i and terminate.

E4. Check the dictionary to see if there are words $w_1, w_2, \ldots, w_m$ ($m \geq 1$) which are similar to S_i. If such words are found, recognize $\bar{X}$ as W_j ($1 \leq j \leq m$) such that w_j is the most similar to S_i and then terminate. {Comment: The measure of similarity between two words is well-described by Hull and Dowling [6]. The criterion chosen for the similiarity measure depends on the user. The criterion chosen for the experiments described in this paper is very simple: Two n-letter words are similar to each other if they differ from each other by exactly 1 letter; e.g. DUMB and DUNB. If $w_1, w_2, \ldots, w_m$ are similar to S_i, then w_j ($1 \leq j \leq m$) is the most similar to S_i if probability of occurrence of w_j in text > probability of occurrence of w_k in text for $1 \leq k \leq m$, $k \neq j$. With this similarity measure, it was found desirable to skip this step for words of length 2 or less, for intuitively obvious reasons.}

E5. Maximize $g(\bar{X},\bar{Z})$ over the n-letter words in the dictionary; that is, recognize $\bar{X}$ as a word $\bar{V}$ if $g(\bar{X},\bar{Z})$ is maximum for $\bar{Z}=\bar{V}$. Then terminate. {Comment: Note this step called the dictionary algorithm(DA). It is executed only if steps E3 and E4 above failed in recognizing $\bar{X}$ for all the d n-letter strings output by the MVA.}.

Thus the above algorithm combines the MVA (Step E1) with a DA (step E5). Let V(n,d) be the complexity of the MVA for a word size of n letters and d alternatives, and let D(n) be the corresponding complexity of DA. The mathematical models of V(n,d) and D(n) are given by Shinghal and Toussaint [14], and so they are not being reproduced here. The mathematical models show that $D(n) >> V(n,d)$. The magnitude of D(n) increases rapidly with increasing size of dictionary. It should be noted that whereas the MVA is executed for all words in the text, the DA is executed for only a fraction β of the words in the text. So the complexity of the hybrid algorithm can be written as $V(n,d)+\beta D(n)$. The value of β is observed experimentally. To reduce the complxity of the hybrid algorithm, it is desirable to have the value of β as low as possible. It should be noted that the lower is β, the fewer are the words in text for which the DA is invoked.

A set of experiments in text-recognition were simulated to observe the performance of the hybrid algorithm. The experiments are described in the following section.

III. THE TEXT RECOGNITION EXPERIMENTS

To conduct the experiments, it was necessary to assemble English language passages to be used as texts for recognition. In this section the assembling of the passages is described first, and then the experiments.

The letters chosen to constitute the passages were taken from Munson's [8] data-set, which contains letters handprinted by 49 writers. The patterns were size-normalized [15] on a 24 by 24 grid G, such that G_{ij} (i-th row and j-th column of G) is equal to 1 for a dark point, and it is equal to 0 for a white point. From each pattern, a feature vector $(x_1, x_2, x_3, \ldots, x_{36})$ was extracted, such that

$$x_k = \sum_{i=a}^{i=a+3} \sum_{j=b}^{j=b+3} G_{ij},$$

where

$$a = 4\left\lfloor \frac{k-1}{6} \right\rfloor + 1, \text{ and } b = 4((k-1) \bmod 6)+1,$$

for $1 \leq k \leq 36$. In the notes accompanying his data-set, Munson identifies two of his writers: one whose letters are "among the cleanest"; the other, "among the most difficult". These two writers shall be respectively called the good and bad writers. It was decided to use the letters written by these two writers as two separate testing sets for the experiments in this paper. This was done to compare the performance of Algorithm E on both well-written and badly written letters. The data-set was then split to exclude the two testing sets, and the recognizer was trained on the remaining letters.

Next, two English passages were assembled to serve as texts to be recognized. The

passages contained the occurrence of only 27 characters: ∦(blank) and the 26 letters from A to Z (all other symbols having been deleted). There was one blank between any two words of a passage. The first passage called the conventional passage comprised 255 words (1096 letters) and was arbitrarily chosen from a newsmagazine. The second passage called the unconventional passage comprised 255 words (1093 letters) and was chosen from Wright's Gadsby [9]. Although the most frequently occurring letter in commonly found English text is the letter E [16] Gadsby is a story of over 50,000 words without a single occurrence of that letter. With the aid of the Chi-square test, Siddiqui [15] has shown that the statistical distribution of letters in the unconventional passage is significantly different from the distribution of letters in commonly found English text. The objective behind selecting the two passages of these natures was to compare the performance of Algorithm E on these passages, when however the statistical information used by the algorithm had been extimated from a corpora of commonly found English text. Thus the passages assembled were these: the conventional and the unconventional passages using the letters written by the good writer, and the same two passages using the letters written by the bad writer.

Three experiments (one each with the MVA, the DA, and the other with Algorithm E) were conducted, using the assembled passages as texts to be recognized. The experiments are described below. The recognition rates observed from the experiments are given in Table 1. The values of β observed for the Algorithm E experiment are given in Table 2.

EXPERIMENT 1: MVA

The experiment was conducted for d=2,3,4 and 5. Table 1 shows that the recognition rates are generally the highest at d=3. For values of d greater than 3, the change in recognition rates is nominal, if at all. This holds true for both the good and the bad writer.

EXPERIMENT 2: DA

Tables 1 shows that the DA gives much higher recognition rates than the MVA. The highest recognition rates are 99.91% (letters) and 99.61% (words) for the conventional passage written by the good writer. The lowest recognition rates are 96.07% (letters) and 92.55% (words) for the unconventional passage written by the bad writer.

EXPERIMENT 3: Algorithm E

Algorithm E was conducted for d=2,3,4, and 5. Table 1 shows that recognition rates stabilize at d=3. For values of d greater than 3, the change in recognition rates is nominal, if at all. However, the recognition rates of Algorithm E are only marginally different from those of the DA. Table 2 shows that as d increases, the value of β decreases. For the good writer, the value of β at d=5 is less than 0.07; for the bad writer it is less than 0.19. As pointed out at the bottom of Section II, it is desirable to have low β, becuase the lower is the value of β, the lower is the complexity of Algorithm E. Compared to the MVA, Algorithm E displays much better recognition rates, for a given passage and writer.

INPUT SEQUENCE: NOW IF THROUGHOUT CHILDHOOD A BRAIN HAS NO OPPOSITION ...

OUTPUT SEQUENCE: NOW HE THROUGHOUT CHILDHOOD A BRAIN HAS NO OPPOSITION ...

Figure 1. The recognized output for a specimen input string

TABLE 1

PERCENTAGE RECOGNITION RATE FOR PASSAGES

Method Used and Value of d		Unconventional Passage				Conventional Passage			
		Good Writer		Bad Writer		Good Writer		Bad Writer	
		Letter Recog. Rate	Word Recog. Rate	Letter Recog. Rate	Word Recog. Rate	Letter Recog. Rate	Word Recog. Rate	Letter Recog. Rate	Word Recog. Rate
MVA	d=2	80.97	47.06	72.64	34.12	84.40	52.94	75.55	36.08
"	d=3	86.00	58.04	72.00	33.33	87.23	60.78	76.46	36.86
"	d=4	"	"	72.28	34.12	"	"	76.83	38.04
"	d=5	"	"	"	"	"	"	"	"
Algorithm E	d=2	96.98	90.98	95.88	89.80	96.44	91.37	95.71	87.45
"	d=3	97.44	93.33	95.97	87.84	96.72	93.33	94.80	86.67
"	d=4	"	"	95.88	87.84	97.08	"	95.26	87.84
"	d=5	"	"	"	"	"	"	94.98	87.45
Dictionary Method		98.17	96.86	96.07	92.55	99.91	99.61	97.81	93.33

TABLE 2

VALUE OF β OBSERVED FOR ALGORITHM E

Value of d used.	Unconventional Passage		Conventional Passage	
	Good Writer	Bad Writer	Good Writer	Bad Writer
2	0.1686	0.2318	0.1012	0.2196
3	0.0824	0.2078	0.0628	0.1765
4	0.0706	0.1804	0.0471	0.1726
5	0.0667	0.1804	0.0471	0.1686

IV. CONCLUDING REMARKS

It is seen that Algorithm E has a complexity very much lower than the dictionary algorithm, although their performance is nearly the same for both the good and the bad writer and for both the conventional and the unconventional passages written by them. For a given passage, the recognition-rates for the good writer are observed to be higher than for the bad writer. The values of β are consistently lower for the good writer than for the bad writer. Thus the complexity of Algorithm E is lower for the good writer than for the bad writer. Munson has commented that the human recognition rate of the characters in his database is from 0.5% to 5.0%. This compares well to the recognition rate of Algorithm E.

ACKNOWLEDGEMENT

I wish to acknowledge the help received from Mr. Khalid J. Siddiqui in the early stage of this work; namely, for programming the preprocessing and the feature-extraction. He had executed this stage for a different set of experiments described in his Master's thesis [15].

REFERENCES

[1] W.W. Bledsoe and J. Browning, "Pattern Recognition and reading by machine", in Pattern Recognition, L. Uhr, Ed., New York: Wiley, 1966, pp. 301-316.

[2] G. Carlson, "Techniques for Replacing Characters that are Garbled on Input", in 1966 Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 28, Washington, DC: Spartan, 1966, pp. 189-192.

[3] F.J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors", Communications of the ACM, Vol. 7, No.3, pp. 171-176, March 1964.

[4] R.O. Duda and P.E. Hart, "Experiments in the recognition of handprinted text: Part II-context analysis", in 1968 Fall Joint Comput. Conf., AFIPS Conf. Proc., Vol. 33, Washington, DC: Thompson, 1968, pp. 1139-1149.

[5] A.R. Hanson, E.M. Riseman and E. Fischer, "Context in Word Recognition", Pattern Recognition, Vol. 8, pp. 35-45, 1976.

[6] P.A.V. Hull and G.R. Dowling, "Approximate String Matching", Computing Surveys, Vol. 12, No. 4, pp. 381-402, December 1980.

[7] H.L. Morgan, "Spelling Correction in Systems Programs", Communications of the ACM, Vol. 13, No. 2, pp. 90-94, February 1970.

[8] J.H. Munson, "Experiments in the Recognition of Handprinted Text: Part I-Character Recognition", in 1968 Fall Joint Computer Conference, AFIPS Conf. Proc., Vol. 33, Washington, DC: Thompson, 1968, pp. 1139-1149.

[9] E.M. Riseman and R.W. Ehrich, "Contextual Word Recognition Using Binary Digrams", IEEE Transactions on Computers, Vol. C-20, No. 4, pp. 397-403, April 1971.

[10] E.M. Riseman and A.R. Hanson, "A Contextual Post-processing System for Error Correction Using Binary n-grams", IEEE Transactions on Computers, Vol. C-23, No. 5, pp. 480-493, May 1974.

[11] J.L. Peterson, "Computer Programs for Detecting and Correcting Spelling Errors", Communications of the ACM, Vol. 23, No. 12, pp. 676-687, December 1980.

[12] R. Shinghal, "An Experimental Investigation of Four Text Recognition Algorithms", IEEE Transactions on System, Man and Cybernetics (in press).

[13] R. Shinghal and G.T. Toussaint, "Experiments in Text Recognition With the Modified Viterbi Algorithm", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-I, No. 2, pp. 184-193, April 1979.

[14] R. Shinghal and G.T. Toussaint, "A Bottom-up and Top-down Approach to Using Context in Text Recognition", International Journal of Man Machine Studies, Vol. 11, No. 2, pp. 201-212, March 1979.

[15] K.J. Siddiqui, "Machine Recognition of Text Using Contextual Postprocessing", M.Comp. Sci. Thesis, Dept. of Computer Science, Concordia University, Montreal, Canada, 1981.

[16] G.T. Toussaint and R. Shinghal", Tables of Probabilities of Occurrence of Characters, Character-pairs, and Character-triplets in English Text", Technical Report No. SOCS 78.6, School of Computer Science, McGill University, Canada, August 1978.

[17] J.R. Ullman, "A Binary N-gram Technique for Automatic Correction of Substitution, Deletion, Insertion and Reversal Errors in Words", Computer Journal, Vol. 20, No.2, pp. 141-147, 1977.

[18] C.M. Vossler and N.M. Bronston, "The Use of Context for Correcting Garbled English Text", in Proc. ACM 19th Nat. Conf., Aug. 1964, pp. D2.4-1 to D2. 4-13.

[19] E.V. Wright, Gadsby, Los Angeles: Wetzel Publishing Co., 1939.

This research was supported by National Science and Engineering Research Council of Canada under Grant No. A3060.

CAA: A Knowledge Based System with Causal Knowledge to Diagnose Rhythm Disorders in the Heart

Tetsutaro Shibahara, John Mylopoulos, John K. Tsotsos and R. Dominic Covvey

Department of Computer Science
University of Toronto

ABSTRACT

This paper presents a framework for the recognition of repetitive time varying signals such as electrocardiograms using an expert system approach. A system called CAA (Causal Arrhythmia Analyzer) is being developed to realize this framework. The CAA system includes:
a) A stratified knowledge base for the description of the underlying event knowledge of electrophysiology in the heart as well as the observable shape knowledge of morphology of ECG wave inputs, where physiological events are projected into the observable wave-shape domain of ECGs. We use a frame-based semantic network with the use of explicit causal knowledge to describe causative temporal relations among underlying events that emit observable signals like ECGs, and
b) A control structure for the recognition of repetitive time varying signals, which uses causal knowledge to check causal integrity among temporal knowledge units (i.e., events) and also to expect and confirm unseen events. This control structure also utilizes similarity links and other organizational primitives such as is-a and part-of hierarchies for its attention mechanism inherited from the previously developed ALVEN (A Left VENtricle Wall Motion Analysis) system [Tsotsos 1980]. The CAA system is being implemented using a version of PSN (Procedural Semantic Networks), a knowledge representation language that has been developed at the university of Toronto [Levesque & Mylopoulos 1979].

1.0 INTRODUCTION

The main objective of this study is to establish a framework for the recognition of time varying signals of a complex repetitive nature, such as electrocardiograms, using a knowledge engineering approach. To this end, an expert system called CAA (Causal Arrhythmia Analyzer) is being developed to diagnose rhythm disorders (usually called arrhythmias) in electrocardiographic monitoring. We have chosen the arrhythmia analysis problem because It is a domain rich in temporal and causal interrelationships. The arrhythmia recognition problem itself deserves attention because the overall performance of existing ECG programs is at most 80% reliable for abnormal ECGs despite efforts since the late 1950's [Hagan 1979]. We believe that a basic reason for this unreliability is that current systems lack underlying event knowledge to handle the complexity inherent in cardiac rhythms.

To diagnose rhythm disorders of the heart, the events in the underlying cardiac conduction system must be exactly determined from one or more streams of observed bodysurface ECG signals [general reference, Sharmroth 1976]. The final objective of arrhythmia interpretation, therefore, becomes the recognition of the physiological and pathological status of the cardiac conduction system, given one or more surface ECG tracings. To accomplish this, unlike other existing or proposed ECG systems, our system utilizes knowledge of the causal structure of underlying physiological events in the heart (what we call "underlying event knowledge"). Based on this knowledge, the system tries to determine the most likely set (or sets) of underlying events that explain the input wave signals. For this purpose, the system introduces and uses causal links extensively in the underlying event knowledge base where various causal relations and temporal constraints are represented.

The causal links in the CAA system play an essential role in characterizing a complex event concept aggregated from other more basic (component) concepts by giving causal and temporal relationships among the component events. Hence, their role is analogous to that of structural descriptions in a composite structure concept in the spatial domain, which relate the component objects making up the whole structure. Thus, using causal links in an event recognition problem, an aggregated event can be recognized not only by its component events, but also by taking into consideration the integrity conditions implied by the causal relations that the component events participate in.

Therefore, this study pursues (1) an adequate representation for causal knowledge, and also (2) a method which effectively uses causality during the recognition process. More specifically, our goal is to investigate the representation of causalities in a rich temporal domain and to determine a control structure that uses causal and temporal knowledge in an integrated fashion for the expectation and the confirmation of foregoing or following unseen events in time.

Our representational approach to the above goal is to construct a stratified knowledge base that contains knowledge about the waveforms of surface ECG signals, the physiology of the heart that explains abnormalities in events and causalities, and the relationships that may exist between surface ECG signals and abnormalities in the heart. A frame-based semantic network representation is used to describe the CAA knowledge base. The KB evolved from the previously developed ALVEN system.

As for the control structure, ALVEN's basic control structure is also used and extended to include the mechanisms which facilitate; (1) the projection from underlying events to observable shapes and vice versa, (2) the expectation and the confirmation of unseen events from known events using causal links, and (3) the recognition of repetitive event sequences with beat-to-beat relationships.

From the AI viewpoint, therefore, the CAA system is considered as an empirical semantic network system for event sequence recognition, which includes (1) the explicit description and use of causative temporal knowledge, and (2) the use of a stratified knowledge base structure with inter-related distinct KBs.

A prototype CAA system is being implemented using a version of PSN (Procedural Semantic Networks), a knowledge representation language that has been developed at the University of Toronto [Levesque & Mylopoulos 1979]. PSN/2 (PSN level 2) now includes similarity links and IS-A/PART-OF hierarchies in addition to the basic primitives such as classes, metaclasses, relations, functions, and programs [Schneider 1978], [Kramer 1980].

2. The Domain of Electrocardiology

Abnormalities in heart function can be viewed as characteristic deviations in observable variables such as blood pressure curves and electrocardiograms (ECGs). The ECG is the record of variations in electrical potential generated by the heart muscle and projected onto the bodysurface.

The ECG tracing is typically composed of a series of waves designated as the P, Q, R, S, and T waves, and a series of segments and intervals between these waves. The Q, R and S waves are often referred to as a group and called the QRS complex. Fig-1 shows a typical standard series of waves for one cycle of heart beat.

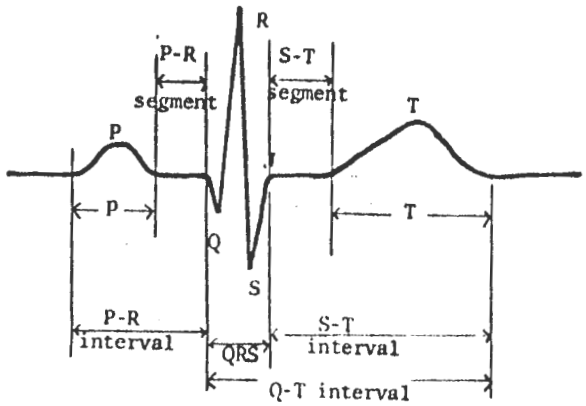
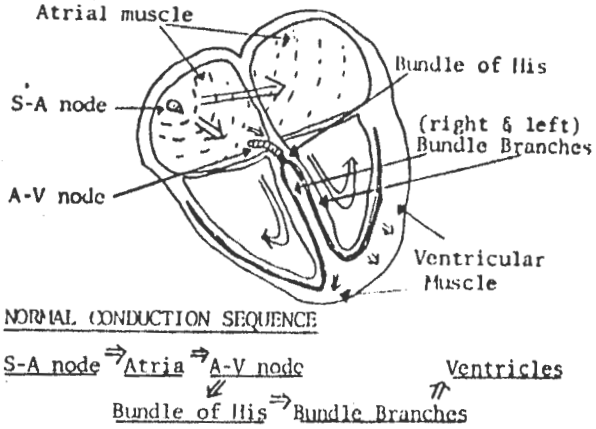


Fig-1 Typical ECG Tracing for One Cardiac Cycle

The underlying physiological origin of the ECG is the cardiac conduction system that consists of the sinoatrial (S-A) node, the atria, the atrioventricular (A-V) node, the bundle of His, the right and the left bundle branches and the ventricular muscle, where the cyclic pacemaking impulse normally originates in the S-A node and ends up at the ventricles as seen in Fig-2.



.Fig-2 The Cardiac Conduction System

The P wave and the QRS complex represent the depolarization phases (i.e., the initiation of muscular contraction) of heart muscle cells in the atria and the ventricles, respectively, while the T wave represents the repolarization phase (i.e.,

the recovery of the cell to its resting state) in the ventricles. No electrical expression is usually seen from the bodysurface for the activities of the S-A node, the A-V node, the bundle of His and the bundle branches.

Arrhythmias (disorders of the cardiac conduction sequence or rhythm of cardiac events) are caused by irregularities in impulse formation or conduction in the cardiac conduction system. The difficulty in the computerized detection and elucidation of most arrhythmias in ECGs lies basically in the wave identification problem.

Some major complications of our recognition problem may be listed as follows,
(1) some kinds of signal shapes are not always identifiable due to low amplitude, overlapping, deformation, noise, etc., i.e., the shape itself of a wave is not sufficient to identify it, rather the context among other waves is important,
(2) shape abnormalities and event abnormalities do not uniquely correspond with each other, rather they represent a many to many mapping across the two domains, and
(3) time varying repetitive behaviors of the model such as progressive shortening of event durations must be accounted for in an extended temporal context.

As will be seen, the first is handled by the use of expectation process about event locations that relies on causal links between events; the segmentation of the system's knowledge base into the underlying event KB and the observable wave-shape KB alleviates the second mapping problem; the recursive class definitions with causal links address the third complication.

3. REPRESENTATION OF KNOWLEDGE

3.1 Types of Domain Knowledge

To represent the diagnostic knowledge of electrocardiography, several distinct types of domain knowledge can be discerned:
a) knowledge about the physiology that explains abnormalities in the heart (underlying event knowledge),
b) knowledge about the form of surface ECG signals (observable shape knowledge), and
c) knowledge about human and machine interventions such as additional measurements and drug administration (strategic knowledge).
Our representational approach to the problem of building such a system as CAA is to construct a stratified knowledge base which contains an independent KB for each type of domain knowledge, with the relationships between concepts across different KBs.

Among the KBs in such a system, the physiological KB is particularly noteworthy since any abnormality in the heart must be explained as a causal phenomenon at the level of the cardiac conduction system, although most of the conduction system's behavior is not observable in the ECG. This physiological KB plays the role of describing not only normal and abnormal events in the heart, (each of which represents the activity of a specific part of the heart), but also the interrelationships among those events such as causal dependencies and temporal constraints. Examining the features of the physiological KB, two causal aspects of the proposed system have been found to be of importance:
(1) the representation of causal links among events to naturally represent complex causal interrelationships among physiological events, and
(2) the estimation of non-observable event parameters such as locations and durations by using the context in which they can be related to observable event parameters through causal links.

3.2 Elements of the Representation

The representation used in the CAA system is a frame-based semantic network representation. Thus, the knowledge units defined in frames are called classes and used to abstract various concepts at each knowledge domain, e.g., class CELL-CYCLE of each portion of the heart muscle, class BEATing, and class BEAT-PATTERN of consecutive heart beats.

Each instantiation of the above class concepts generates a class-token or simply token; thus, a token "normal-beat of John" is an instance of the class NORMAL-BEAT.

The above instance-of relation is extended to a relation between a class and a meta-class. For example, any statistical data about class BEAT itself cannot be the attributes of any specific beat instance. This is a rather important distinction that most medical expert system do not make: statistical information, so commonly used in medical diagnosis systems, should be represented as attributes of the class rather than of instances of the class. Furthermore, this separation can be viewed as a default mechanism. In disease classes for which insufficient information is known to diagnose them categorically [Szolovitz & Pauker 1978], statistical information are usually contained either in the definition of the class, or with respect to a particular patient case. In such cases, however, metaclasses may be defined and used as the default reasoning mechanism.

A class-frame consists of three kinds of descriptors: the component descriptor, the organizational descriptor, and the link descriptor. The component descriptor is made up of slots filled with component (part) classes. The organizational

descriptor includes is-a and instance-of phrases, which indicate the corresponding parental concepts in these respective organizational relations. The link descriptor includes two types of link information: similarity links and causal links. Similarity links are associated with component classes and used as an aid in activating alternative parallel hypotheses when exceptions occur in the recognition of these components [Tsotsos 1981].

3.3 Representation of Causal Links

Rieger's CSA (Common Sense Algorithm) system [Rieger and Grinsberg 1976 & 1977] and Patil's ABEL system [Patil et al 1981] are perhaps the best examples of current causal representations. Rieger and Grinsberg introduced several types of causal links for the Mechanism Lab in the Common Sense Algorithm system, and also distinguished two types of causal flows: continuous causality and oneshot causality. Although their syntactic event classification is not applicable to our system, we adopted the ideas of continuous and oneshot causal flows, and the idea of gating conditions. Patil and others introduced a multi-level causal network to explain aggregated structures in diseases such as diarrhea. Although the ABEL system explains the aggregating process from basic physiological causal links to more global causal links between disorder events within one disease, it does not seem to provide the causal links classified by the types of influence and temporality. These latter concepts are essential to describe time varying phenomena as seen in the electrophysiology of the cardiac conduction system.

3.3.1 Features in a Causal Link

We regard a causal link as a representation of causality in which we may observe some flow of influence between two distinguishable events. We may characterize a causal link at least by the following two features:
(1) the existential dependency of an effect event on its cause event(s), i.e., the feature that no effect events can exist or happen without cause events, and
(2) the temporal constraints between the cause events and the effect events, i.e., the feature that the former must precede the latter in time, with a possible delay time interval.

We should be aware that while a causal link implicitly includes a temporal constraint, the existence of a temporal constraint does not necessarily mean a cause-effect relationship between two events, but it may strongly suggest the existence of a certain underlying chain of causal links.

As well as temporal constraints, there could be other associated constraints with a causal link. Most typical constraints of this kind are boundary conditions on any state describing variable such as temperature, pressure or the potential at the boundary time point between two consecutive events.

3.3.2 Types of Causal Links

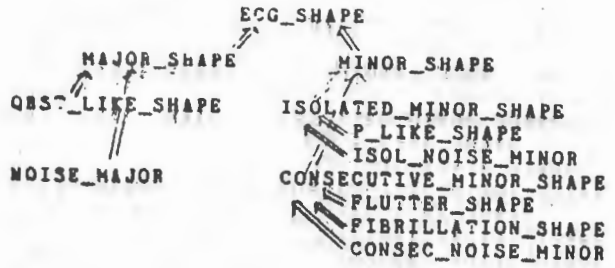
Causal links of the CAA system are classified according to (1) types of influence, and (2) types of temporal constraints. The type of influence of a causal link must be defined by the role of the link, the type of dependency, and the roles of participating events such as cause, effect, and condition. The type of temporal constraints in a causal link is usually understood implicitly from the meaning of the link, which implies the temporal relationships between the participating events in the link. In addition, associated constraints on state variables may be attached to these causal links.

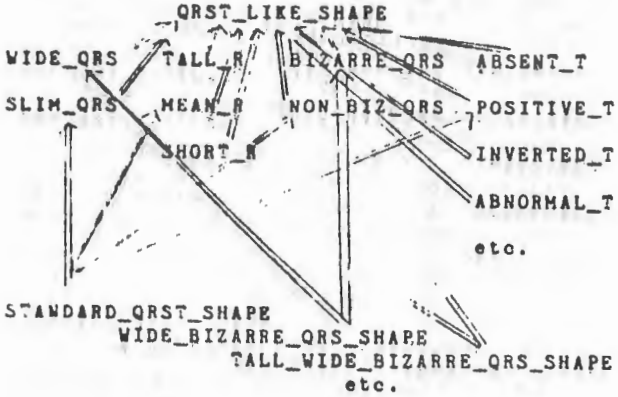
Some useful one-shot type causal links in CAA are the following:
(1) TRANSFER, TRANSITION -- causal links which describe state (or phase) change from the preceeding event to the following event in time involving a single subject; TRANSFER indicates the subject normally completes the preceeding state (event) and changes into the following state; and TRANSITION means the subject is forced to terminate the current state and transition into a new state.
(2) INITIATION, INTERRUPT -- causal links in which a causative event of one subject initiates or interrupts an effect event of another subject; In INITIATION, a causative starting (or ending) event of one subject triggers a new event of another subject, and in INTERRUPT, a causative event of one subject interrupts (and forces to terminate) an event of another subject and make it transition to a new state.

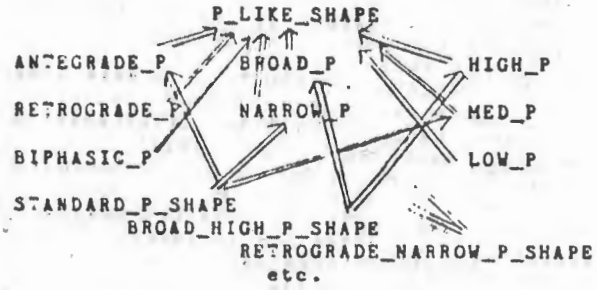
Note that since the above causal links are one-shot type, the causal flow occurs only once at the starting (or ending) time of the causative event to cease/start the other events.

3.4 Examples of Domain Knowledge Representation

The first example is a portion of the IS-A hierarchy from the wave-shape KB of CAA, to show how a generic (shape) concept can be specialized into a specific concept along this hierarchy.







An example using the part-of structure of concepts in the physiological domain follows. The underlying event KB contains the following generic concepts each of which corresponds to each description level (part level) of the physiological knowledge. These concepts are:
(1) the CELL_PHASE generic concept describes each phase of cell activity and is temporally a part of a CELL_CYCLE concept;
(2) the CELL_CYCLE generic concept indicates the cell activity of a small portion of a part of the heart and becomes a part of an ACTIVITY concept;
(3) The ACTIVITY generic concept represents the activity of one part of the heart such as the atrium and the A-V node and takes part in a BEAT concept;
(4) the BEAT generic concept describes a single conduction sequence of the heart and is a component of a BEAT_PATTERN concept;
(5) the BEAT_PATTERN generic concept describes the basic segment of a cardiac rhythm and is a part of a PATTERN_SERIES concept; and
(6) the PATTERN_SERIES concept describes a periodic series of rhythm segments (BEAT_PATTERNs).

Using specialized concepts from the above generic concepts, we are able to represent any specific physiological event in our class-frame formalism. For example, the following class definition represents a normal conduction sequence in the heart. (The dot "." notation in a slot means the part concept specification as used in ALVEN.)

```
class SINUS_PACING_BEAT
with
components
  sanode-cycle:
    SAN_MATURE_CELL_CYCLE;
  atrium-activity:
    ATR_MATURE_FORWARD_ACTIVITY;
  avnode-activity:
    AVN_MATURE_FORWARD_ACTIVITY;
  lv-activity:
    LV_MATURE_FORWARD_ACTIVITY;
  av-interval: .............;
    /* other component definitions
          follow here..............*/

causal-links
  sanode-atrium-propagation: INITIATION
    causative-ending-event:
     sanode-cycle.depolarization;
    starting-event:
     atrium-activity.upper-cell-cycle.
      depolarization;;

  atrium-avnode-propagation: INITIATION
    causative-ending-event:
     atrium-activity.lower-cell-cycle.
      depolarization;
    starting-event:
     avnode-activity.upper-cell-cycle.
      depolarization;;

  avnode-lv-propagation: INITIATION
    /* similar to the above */;;

end
```

The above causal links give the causative sequence of impulse propagation, where the depolarization phase of one part of the heart triggers the depolarization phase of the next part.

The class names used in the slots of the above class frame must be defined independently elsewhere, such as,

```
class SAN_MATURE_FORWARD_CELL_CYCLE
with
components
  depolarization:
    SAN_MATURE_DEPOL_CELL_PHASE;
  under-repolarization:
    SAN_MATURE_UNDER_REP_CELL_PHASE;
  partial-repolarization:
    SAN_MATURE_PART_REP_CELL_PHASE;
  complete-repolarization:
    SAN_MATURE_COMP_REP_CELL_PHASE;
causal-links
  transfer-from-depol-to-repol: TRANSFER
   ending-event: depolarization;
   starting-event: under-repolarization;

  transfer-from-under-repol-to-part-repol:
  TRANSFER
    /* similar to the above */
  transfer-from-part-repol-to-comp-repol:
  TRANSFER
    /* similar to the above */;;

end
```

In the above causal links, a single subject (SAN: S-A node) is changing its state from one phase to another using TRANSFER links.

The above definition also shows a CELL_CYCLE consists of four kinds of CELL_PHASEs: a depolarization cell-phase, an under repolization cell-phase, a partial repolarization cell-phase, and a complete repolization cell-phase.

4.0 CONTROL STRATEGY FOR RECOGNITION

Based on the ALVEN's control structure in general, the control structure of the CAA system has been extended and developed for three purposes:
(1) to exploit causal knowledge about events,
(2) to provide means of communication across distinct KBs, and
(3) to recognize repetitive event sequences and to detect beat to beat relationships.

4.1 Expectation and Confirmation through Causal Links

The task we are considering is the recognition of complex time-varying events. The role of causal relationships in such events is to provide local context for their components or constituents, i.e., it is to produce expectations of the properties of this aggregated event backward or forward in time. The system , therefore, must look ahead or look back for these causally linked component events, starting with one or more already-identified component events. Thus, causal links are used to locate the temporal positions of "to-be-observed" events. That is to say, if there are events that are causally linked, we can generate the probable locations of intermediate and terminal component events by looking forward or backward through causal links from the locations of known events.

In the above process the linked component events sometimes, however, are not observable as waveforms in the input stream. In such a case, the system will supply these non-observable variables (such as event durations) with appropriately estimated values using the local context of the event. For this purpose, the statistical standard values must be defined through the metaclass of each (component) event. Since the metaclass defines the properties of the class concept itself such as statistical and default values, the system refers to this kind of metaclass knowledge when it faces lack of information in the recognition process.

On the other hand, the temporal locations and the shapes of the expected events must be confirmed if these events have observable counterparts in the shape domain. This is the confimation process from the event domain to the shape (signal) domain.

4.2 Recognition Steps - Initial Stage

Signal recognition starts with picking up some prominent shapes in an input signal stream, which we name anchoring shapes. Anchoring shapes can be easily found in our ECG domain by finding the steepest slopes in the signal, namely QRS complexes. In general, the anchoring shapes for a specific signal recognition problem must be classified and described. For this purpose, we have defined the wave-shape KB which contains various morphologies of input waveforms together with their appropriate components and generalizations. As seen in the first example of section 3.3, the wave-shape KB defines class MAJOR_SHAPE as the anchoring shape; thus, it becomes the IS-A parent class of shape class QRST_LIKE_SHAPE and the IS-A ancester class of class WIDE_BIZARRE_QRS_SHAPE. In this fashion, in the wave-shape KB, knowledge classes are organized via the PART-OF and IS-A relations.

We will examine the process of this recognition in more detail in the following steps.

-Initial Shape Analysis

When a series of signal tokens are given to the recognition system, the system first tries to find an anchoring shape to start the recognition with. Suppose we have chosen QRST_LIKE_SHAPE for the anchoring shape. As the first step, the system picks up a certain starting set of signal tokens. Gradually posing constraints to the generic QRST_LIKE_SHAPE

class. the system tries to identify the token set to be a specialized QRST complex wave shapes such as STANDARD_QRST_SHAPE and WIDE_NON_BIZARRE_QRST_SHAPE. The focus and change of attention mechanism of the ALVEN system is used to find the best matching shape hypotheses to the anchoring shape in the signal domain (Tsotsos 1981).

-Association Process from the Observable Wave-shape KB to the Underlying Event KB

Once the shape hypotheses are reached (specialized QRST complex waves), the system must look for the source events of these shapes, which are non-observable in the signal domain. This underlying event seeking process associates each shape concept with its several underlying event classes. This association across the two domains can be realized in the following fashion. Suppose we have a shape hypothesis with a standard QRS duration such as STANDARD_QRST_SHAPE. Since we know either a normal left ventricular (LV) activity, a self decay LV activity (escape), or a self decay in the bundle of His, may generate the normal QRS shape and duration, we define the association path as

```
STANDARD_QRST_SHAPE
  |--> LV_MATURE_FORWARD_ACTIVITY
  |--> LV_MATURE_FOCUS_UPLV_ACTIVITY
  |--> LV_MATURE_COMPLETE_DECAY_ACTIVITY
```

Similarly, if the shape has a wide and non-bizarre QRS complex, it will be associated with

LV_PREMATURE_FORWARD_ACTIVITY, and
LV_PREMATURE_FOCUS_UPLV_ACTIVITY.

The basic idea of this association is to maintain the list of the events which project this recognized shape as the counterpart in the signal domain.

-Exploring Probable Global Events from a Local Event

After associations have activated physiological event hypotheses, each event hypothesis tries to find a set of global event classes each of which possesses the original event hypothesis as a component event. We call this process the exploration of global events. These explored global events such as various BEAT events are the events to be hypothesized next.

One example is the exploration

```
from LV_MATURE_FORWARD_ACTIVITY
to (1) SINUS_PACING_BEAT (normal)
   (2)    PREMATURE_ATRIAL_BEAT    (atrial
focus)
   (3) AV_NODAL_BEAT (A-V nodal focus)
      ..........
```

-Expectation and Confirmation Processes Using Causal Links

Now that we have explored probable BEAT events, for each BEAT event, we must try to check the credibility of these events by instantiating their component events (mainly ACTIVITY events for parts of the heart). This instantiation starts with the known events which were associated with the established shape hypotheses since the starting and the ending times can be determined from those of the shape hypotheses. Since causal links include temporal constraints between causally related component events, the system can estimate the times of "to-be-observed" events from the known events using these causal relationships. We assume that average durations of events are given through their metaclasses; therefore, the system can easily expect the probable temporal locations of all the events which are causally related to the known events. We call this process the expectation process.

The locations and the shapes of the expected events must be confirmed if these events have observable counterparts in the signal (shape) domain. This confirmation process from the event domain to the signal shape domain is similar to the previous association process which works in the opposite direction. If one (or more) of the confirmation processes fails, the corresponding global hypothesis fails and must be withdrawn.

In ECG recognition, the location and the shape of the P wave must be expected and confirmed, both of which are very different according to the explored BEAT hypotheses.

-Final Beat Hypotheses at the Initiation Stage

We have a set of eligible BEAT hypotheses at the end of the initiation stage, each of which has passed the criteria on the matching and integrity scores (certainty factor) about each BEAT class.

4.3 Repetitive Beat Recognition Stage

To diagnose a arrhythmia, its repetitive behavior in a series of BEATs must be recursively defined. In the following simplified example, a repetitive pattern of (imaginary) SLOW_BEATs is defined in three class frames.

Note that the recursion is done through class SLOW_BEAT_CYCLES, which is the IS-A parent of the last two repetition classes.

```
class SLOW_BEAT_PATTERN
with
  components
  initial-beat: NORMAL_SINUS_BEAT;
  beat-cycles:  SLOW_BEAT_CYCLES;
  final-beat:   NORMAL_SINUS_BEAT;
end
```

```
class  SLOW_BEAT_REPETITION
  is-a SLOW_BEAT_CYCLES
with
  components
  first-beat: SLOW_BEAT;
  successive-beat: SLOW_BEAT_CYCLES;
end

class SLOW_BEAT_REPETITION_UNIT
  is-a SLOW_BEAT_CYCLES
with
  components
  first-beat: SLOW_BEAT;
end
```

Once beat patterns have been defined in the above fashion, the recognition starts with one of the beat hypotheses established at the previous initiation stage. Examining this first hypothesis, the system hypothesizes several beat pattern classes that possess the same beat class in their definitions. While the system recursively generates successive beat classes, the recognition proceeds one beat to another along the time axis repeating the expectation and the confirmation processes. In this recognition, similarity links are essential in the sense that the similarity links between repetitive beat patterns enable the hypothesis competition and cooperation mechanism to work along with the progress of time. Also, the causal links between consecutive BEAT classes enable the system to verify the causal relationships among corresponding components on a beat-to-beat basis and also estimate the periodicity of a series of beats as the whole.

The final arrhythmia interpretation hypotheses will be determined as those beat patterns that passed the criterion on the overall matching scores calculated in the above process.

5.0 CONCLUSIONS

The basic design of the CAA system has been completed and its implementation is underway using the PSN/2 language. We have shown that the inclusion of causal links in a frame-based semantic network, along with the organizational primitives IS-A and PART-OF, has allowed us to tackle the problem of reconstructing complex electrophysiological event sequences from gross signal characteristics. This is accomplished by defining the semantics of causality and noting that it is these semantics that can be used for the generation of expected signal characteristics and other associated events. More generally, the inclusion of causal knowledge provides a context for the recognition and reconstruction of complex event sequences.

REFERENCES

[Hagen, A. et al 1979] "Evaluation of Computer Programs for Clinical Electrocardiography", Computer Techniques in Cardiology, Cady Jr., D. (ed.), Marcel Dekker Inc., New York, 1979

[Kramer, B.M. 1980] "The Representation of Programs in the Procedural Semantic Network Formalism", Tech. Rep. CSRG-116, Computer Systems Research Group, Univ. of Toronto, August 1980

[Levesque, H. and Mylopoulos, J. 1979] "A Procedural Semantics for Semantic Networks", Associative Networks: Representation and Use of Knowledge by Computers, Findler, N.V. (ed.), Associated Press, New York 1979

[Patil, R.S. et al. 1981] "Causal Understanding of Patient Illness in Medical Diagnosis", IJCAI 1981

[Rieger, C. and Grinsberg, M. 1976] "The Causal Representation and Simulation of Physical Mechanics", Tech. Rep. 495, Univ. of Maryland, Nov. 1976

[Rieger, C. and Grinsberg, M. 1977] "The Declarative Representation and Procedural Simulation of Causality in Physical Mechanics", IJCAI 1977

[Schneider, P.F. 1978] "Organization of Knowledge for a Procedural Semantic Network Formalism", 2nd CSCSI, Toronto, July 1978

[Szolovitz, P., & Pauker, S.G. 1978] "Categorical and Probabilistic Reasoning in Medical Diagnosis", Artificial Intelligence, 1978, 11, p.115-144

[Tsotsos, J.K. et al. 1980] "A Framework for Visual Motion understanding", IEEE Trans. of Pattern Analysis and Machine Intelligence, PAMI-2, No.6, Nov. 1980

[Tsotsos, J.K. 1981] "Temporal Event Recognition: An Application to Left Ventricular Performance", IJCAI 1981

USING REACH DESCRIPTIONS TO POSITION KINEMATIC CHAINS

James U. Korein

Dept. of Computer and Information Science
Moore School
University of Pennsylvania
Philidelphia, PA 19104

Abstract

A kinematic chain with redundant degrees of freedom and joint limits is to be adjusted to bring its distal end to a specified position. For example, the distal end of a human arm or robot manipulator is to be positioned. An algorithm which uses perfect descriptions of reach, (workspaces), is presented, and shown to be correct. The constraint that the reach descriptions be perfect is then relaxed, and the algorithm is seen to work for a certain class of conservative approximations.

I. Introduction

A chain is a sequence of rigid links, connected by joints. One end of a chain is designated the proximal end. The proximal end of the proximal link is connected, by a joint, to a reference link fixed in a world coordinate system. The pedestal of a manipulator is an example of a reference link. The other end of the chain, called the distal end, is free to move in space. In this paper we consider the problem of positioning the distal end of a chain, Chi.

One application is in graphic modelling and evaluation of human environments, such as workstations or cockpits. A similar problem arises in the control of robot manipulators. In this case, however, the problem is usually to achieve a specified position and orientation for the distal link. Human beings and other chordates must also solve this problem repeatedly in their normal activities.

The degrees of freedom of a chain correspond to a set of independent variables describing the ranges of motion permitted by the joints. The range of values which may be taken on by each joint variable is usually limited. "Allowable" values for joint variables and "allowable" configurations are those which do not violate these limits. Independence of joint variables implies that the joint limits for one variable do not depend on the value of any other.

This work was supported in part by NSF grant number MCS-078-07466.

Definition: The workspace of a chain, Chi, is the set of all points in space which may be reached by the distal end.

The dimensionality of the workspace depends on the number of joint variables of the chain, and the relationship between succesive joints. If Chi has n degrees of freedom, its workspace is an object with dimensionality no greater than n. When the number of degrees of freedom is the same as the dimensionality of the workspace, the chain is non-redundant.

In the non-redundant case, there are only a small number of configurations of the chain which will position the distal terminal at any point in the workspace. For a particular point, these configurations may be obtained by solving a system of n equations in n unknowns*, ($n \leq 3$), or by application of fundamental geometric principles [7,8]. Since the algebraic and geometric procedures for obtaining values for joint variables do not account for joint limits, it may be necessary to discard configurations which are not allowable. By definition, the workspace is the set of points which can be reached by at least one allowable configuration.

If the number of degrees of freedom of a chain is greater than the dimenaionality of its workspace, the chain is redundant. This is always the case when there are more than three degrees of freedom, since the workspace is confined to physical space. A redundant chain may reach most points in its workspace with any of an uncountable number of configurations. This is easy to see, since a small adjustment to one joint variable may usually be compensated for by the others.

Redundant chains are of practical interest for several reasons. External constraints, such as the avoidance of obstacles, may disallow the few configurations in which a non-redundant manipulator can reach a particular point. Also, redundancy allows for a small change to be made in the positon of the distal terminal by comparably small changes in the chain degrees of freedom. This is not necessarily the case for a

* Typically, the equations relate the coordinates of the distal end with the joint variables, which are unknown.

non-redundant chain with joint limits.

II. Configuring Redundant Chains

Given a chain with n degrees of freedom and its initial configuration, what adjustments should be made to each degree of freedom in order to position the distal end in a specified goal position?

There are three equality constraints on this system, arising from the triple of coordinates used to define the goal position. If n, the number of degrees of freedom of the manipulator, is greater than three, the system is underconstrained.

One approach to this problem is to attempt to find a solution using only three of the n degrees of freedom. A generalization of this approach, suggested by Whitney [18] is to manufacture n-3 additional equality constraints, to obtain a perfectly constrained system. For example, the position of one joint, relative to another, might be specified. However, the imposition of permanent constraints limits the capabilities of the manipulator, essentially defeating the purpose of redundancy.

Another method is to introduce an optimization criterion, and to formulate the problem as a constrained optimization problem [1,5]. This problem has three non-linear equality constraints and 2n linear inequality constraints, arising from upper and lower limits on each of the n degrees of freedom. If joint-limit constraints are ignored, it is possible to attack the problem with the method of Lagrange Multipliers [18]. This method produces a perfectly constrained system of equations, which, in general, must be solved numerically. It is then necessary to guarantee that the iterative process converges to a position satisfying the joint limit constraints. One possible method of doing this is by using a penalty function, which increases the value of the objective function as joint variable values approach their limit [4].

We propose a different approach, which uses information about the workspace of the chain and some of its subchains. It will be assumed initially that perfect descriptions of these workspaces are available. The implications of using less than perfect descriptions will be examined subsequently.

III. An Algorithm for Adjusting Redundant Chains

Let us impose a linear ordering on the joint variables of the manipulator, and label them q[1] through q[n]. For simplicity, let us suppose that each joint has one degree of freedom, and number them from the most proximal, to the most distal. In this case we can label each joint i with the name of its joint variable, q[i]. For consistency, we label the distal terminal q[i+1]. An example is shown in figure 1.

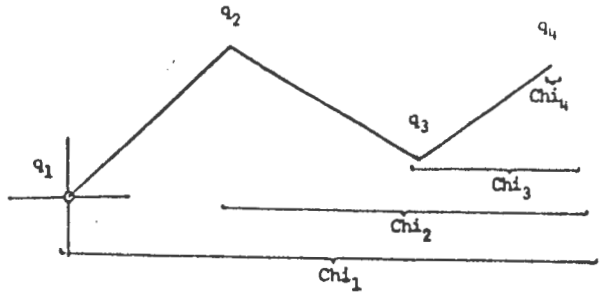


Figure 1

We will call the chain Chi[1]. The distal subchain beginning with joint q[2], and extending to the distal end is called Chi[2]. The next one, beginning at q[3], is Chi[3], etc., up to the degenerate distal subchain Chi[n+1] with zero degrees of freedom. The workspace of each chain Chi[n] is called W[n].

Algorithm Reach

1. If the goal point is not in the workspace of the chain, W[1], then halt immediately and report failure.

2. Otherwise, adjust the first joint variable, q[1], enough to bring the goal point within W[2], the workspace of the next subchain. It may be the case that no adjustment is necessary, if the goal point is already within W[2].

3. Repeat the last step for each remaining degree of freedom, q[i], for each i up to n, inclusive. Each adjustment effectively brings the goal within reach of the next subchain, Chi[i+1].

The algorithm will fail in step 1 if and only if the goal point is unreachable by the chain. In step 2, in order for the necessary adjustments to be made, it must be the case that there is always an allowable value for the joint variable to be adjusted that will bring the goal within reach of the next subchain. That is:

Theorem 1: If a point p is in W[i], then there is some allowable value of q[i] which will bring p into W[i+1].

Proof: If the point p is within W[i], for i in [1,n] then it is, by definition of the workspace, reachable by Chi[i], in some allowable configuration. Suppose that the value of the joint variables in this configuration are v[i], v[i+1], ..., v[n] for variables q[i], q[i+1], ..., q[n], respectively. Then, when q[i]=v[i], and the joint variables of subchain Chi[i+1] take on values v[i+1],...,v[n], respectively, its distal end will be at point p. But since p is reachable

by Chi[i+1], it is in W[i+1].||

The procedure for adjusting a joint variable to bring the goal point within reach of the next subchain depends upon the nature of the joint. The situation, for a revolute joint, is depicted in figure 2.

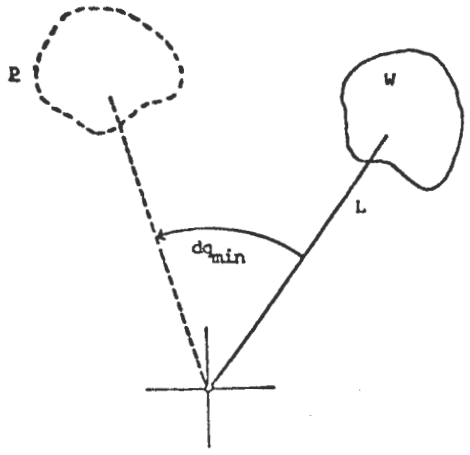


Figure 2

Workspace W is rigidly affixed to the end of link L. The joint variable, q, must be adjusted by some amount dq so that the workspace W envelopes the goal point p. Let the minimum and maximum values for dq, (which put p on the boundary of W), be dq[min] and dq[max]. The problem is simplified by noting that there is a dual problem, shown in figure 3.

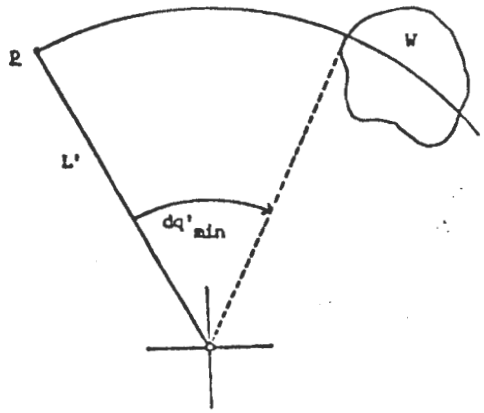


Figure 3

Here, we suppose that p is rigidly affixed to the end of an imaginary link L', and that W is fixed. Now consider the adjustment, dq', of L', necessary to bring it into the fixed workspace. Let the minimum and maximum adjusments be dq'[min] and dq'[max]. We see that dq'[min] has the same magnitude as dq[min], but the opposite sense, and likewise for dq'[max] and dq[max]. Moreover, we see that as the "link" L' moves, p sweeps out a circular arc. Thus the dual problem, and hence, the original problem, may be solved by finding the intersections between this arc and the boundary of the workspace. The values of dq at which the intersections occur are the extremes of the range of values of dq which bring the goal into the workspace. If the workspace has concavities, several such ranges may exist. Only ranges, or parts of ranges which satisfy joint limit restrictions are allowable.

The adjustment problem is simpler for sliding joints. In the dual to this problem, the goal point p sweeps out a line segment rather than an arc. The intersections between this segment and the workspace boundary then give the extreme values for the adjustment. For both revolute and sliding joint cases, the solution to the intersection problem depends on the way the workspace is represented.

IV. Approximate Descriptions of the Workspace

No method for computing the exact workspace of an arbitrary chain with joint limits has been published at this time [3,5,10,11]. One apprxoximate method has been proposed by Kumar [10,11] and Derby [3] independently. A similar approach was proposed by Sugimoto and Duffy [16,17]. Points on the boundary of the workspace for a manipulator with three or more degrees of freedom are computed by finding the maximum extension in a specified direction. By varying the direction incrementally, a "shell" of points which lie on the boundary may be found.

The algorithm proposed in the last section presupposed perfect descriptions of the workspace of each chain Chi[1], ..., Chi[n]. What is the consequence of using approximations?

In general, the consequence is that Theorem 1 no longer holds, and the algorithm will not work. Let W[i] be an approximation to workspace W[i]. It can no longer be guaranteed, in all cases, that when a point is in W[i], there will be some adjustment to q[i] will bring it into W[i+1].

Suppose, however, that the workspace approximations are constructed in such a way so as to guarantee that Theorem 1 holds, for every succesive pair, W[i] and W[i+1]. Then, once a workspace approximation which contains the point is found, the algorithm will proceed to the end.

The approximation to the last workspace, W[n+1], had better be conservative, (that is, a subset of W[n+1]). Otherwise, when we get to the end, it may not be possible to position the distal end properly. This requirement, in conjunction with the requirement that the theorem be satified, implies that, for each i, the approximation W[i] must be conservative. The requirement that all approximations W[i] be conservative is necessary, but not sufficient. For example, we could even construct conservative approximations W[i] and

W[i+1] which are disjoint.

Now, since we begin the algorithm with the approximate workspace W[1], rather that W[1], the algorithm will immediately indicate failure for a goal point belonging to W[1] but not to W[1], as well as for points not in W[1]. The algorithm will, however, produce a correct solution for any goal point in W[1].

In summary, algorithm Reach will work with any set of conservative workspace approximations W1[1], ..., W[n+1] whose successive pairs satisfy Theorem 1. The algorithm will work only for points in W[1].

A full discussion of the construction of workspaces is beyond the scope of this paper. However, we will make a few points relating to the requirements established above.

Consider first workspaces of dimensionality less than three. Since these workspaces exist in three dimensional space, they consist entirely of boundary points. For example, a two dimensional workspace is a surface patch. The only conservative approximations to that patch are its subsets. But obtaining such an approximation is typically no simpler than obtaining an exact description.* We conclude that the concept of conservative approximation is not useful for workspaces of lower dimensionality.

It is not until the workspaces reach the dimensionality of the space in which they are imbedded that they come to have interiors. In this case a conservative approximation is any set, all of whose points are interior to or on the boundary of the workspace. For example, a polyhedron enclosed in the workspace might serve as a useful conservative approximation.

The methods for approximating workspace discussed earlier [3,10,11] are obtained by maximally extending the chain in many different directions. The immediate temptation is to join each surface point found in this way to its nearest neighbors by some triangulation scheme, thereby obtaining a polyhedral approximation. But such an approximation is not guaranteed to be conservative. In fact, since workspaces are not necessarily convex objects, it is not immediately apparent how deep a cavity might lie between directions in which the maximum extension is known. Moreover, even if it is possible to "pare down" such an approximation to guarantee its inclusion within the true workspace, the relationship between each pair of succeaive workspace approximations W[i] and W[i+1] is not guaranteed to satisfy the condition required by Theorem 1.

---

* We ignore subsets consisting of scatterings of isolated points or curves. To be useful for our purposes, an approximation should have the same dimensionality as the approximated object.

Consider a different approach to the computation of workspaces. We begin with W[n+1], the workspace of the degenerate chain Chi[n+1], which is a single point. Then, each succesive workspace W[i] is computed on the basis of the last workspace, W[i+1], and the parameters of the joint q[i] and the link which distinguish the two. The parameters most commonly used in the robotics literature are due to Denavit and Hartenberg [2]. We may think of W[i] as the volume swept out by W[i+1] as q[i] varies over its range of allowable values; (see figure 4).

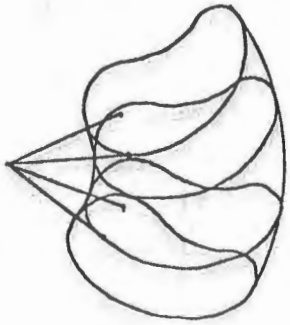


Figure 4

Further discussion of algorithms of this type appears in the author's Ph.D. thesis proposal [9]. Gupta and Roth [6] have applied this approach to manipulators with unrestricted revolute joints, in order to compare workspaces of manipulators with alternative designs. For application to control, this general approach to the computation of workspaces has a major advantage. We compute each workspace approximation W[i] in terms of W[i+1]. If the operation of sweeping W[i+1] can be done conservatively, so that no "extra" space is included, then W[i] and W[i+1] will satisfy the condition required by Theorem 1. Moreover, since we begin with an exact description of W[n+1], which is just a point, then W[n] and all subsequent workspace approximations will be conservative. These are exactly the conditions required for algorithm Reach to work.

V. Extensions

The algorithm Reach was not completely specified, in the following sense. We showed that we could always adjust a joint variable to bring the goal point into the next workspace, but we never stated exactly which adjustment would be chosen. The strategy which comes to mind immediately is to choose the smallest satisfactory adjustment. If this strategy is used, the configuration resulting from the algorithm will have the property that joint variables at the end of the imposed ordering are always favored for adjustment. If the ordering chosen is proximal to distal, this means that no joint will be moved unless the goal cannot be reached with the remaining, more distal portion of the chain.

This strategy may prove useful in several applications. Suppose that a robot consists of a manipulator on a wheeled base. The chain representing the robot includes the degrees of freedom of motion over the floor.* When a goal is to be reached, positioning and orienting the robot, and adjusting its arm are all handled by the same procedure. Consider another application of the algorithm Reach to some chain in a complex linkage with a tree structure, like the human body. The property stated above implies that the reach operation will not alter any parts of the tree that really needn't be disturbed. That is, if an operation caused some link in the reaching chain to be moved, the reach operation could not have been performed without moving it. This is advantageous in the context of performing simultaneous or overlapping tasks.

For a chain with three links, a closed-form solution for joint variables may be evaluated very quickly. For this reason, it may be desirable to use such a solution for the final links of a redundant chain. In this case, the adjustment for previous degrees of freedom may still be found using the algorithm Reach. The last three iterations are simply replaced by a procedure to evaluate the closed-form solution.

The ideas inherent in the algorithm Reach may, in theory, be extended from the purely positional to the position-orientation domain. Consider the problem of positioning the distal link of a manipulator with a specified position and orientation. A specification of position and orientation in space has six degrees of freedom; thus each such specification may be viewed as a vector in a six dimensional position-orientation space. We define the p.o. workspace of a chain as the set of all such vectors attainable by the distal link. Given descriptions of the p.o. workspaces of a chain and its subchains, we could perform a process analogous to the one described for workspaces. In practice, approximate representation of objects of such high dimension tend to be too large to be useful. No descriptions of the p.o. workspace, (as it is defined here), have been published at this time [10,15].

The algorithm Reach exemplifies a general principle. A workspace describes one aspect of the capabilities of a chain. A system can use knowledge about its own capabilities and the capabilities of its subsystems to partition a task in accordance with those capabilities.

VI. Summary

We have presented an algorithm whereby a chain with redundant degrees of freedom and joint limits may reach a goal point with its distal end. A linear ordering is imposed on the degrees of freedom. The algorithm uses information about the workspaces of the chain and its subchains to determine the adjustments necessary for each joint variable, in succesion. It is required that the representations used for workspaces be conservative, and that when a point is in the ith workspace, there is an allowable value for the ith degree of freedom which will bring it into the i+1st workspace.

Acknowledgement

I would like to thank Norman Badler for his guidance and support.

References

[1] L. Cooper and D. Steinberg, Introduction to Methods of Optimization, Philadelphia: W. B. Saunders Company, 1970.

[2] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," Journal of Applied Mechanics, Vol. 77, pp.215-221, June 1955.

[3] S. Derby, "The Maximum Reach of a Revolute Jointed Manipulator," Mechanism and Machine Theory, Vol. 16, pp. 255-261, 1981.

[4] P. E. Gill and W. Murray, Numerical Methods for Constrained Optimization, New York, Academic Press, 1974.

[5] B. Gottfried and J. Weisman, Introduction to Optimization Theory, Englewood Cliffs, N. J.: Prentice Hall, 1973.

[6] K. C. Gupta and B. Roth, "Design Considerations for Manipulator Workspace," ASME Technical Paper No. 81-DET-79, Sept. 1981.

[7] B. K. P. Horn and H. Inoue, "Kinematics of the MIT-AI-VICARM Manipulator," M.I.T. Artificial Intelligence Laboratory Working Paper 69, May 1974.

[8] B. K. P. Horn, "Kinematics, Statics, and Dynamics of Two-Dimensional Manipulators," Artificial Intelligence: An MIT Perspective, Vol. 2, ed. P. H. Winston, Cambridge, Mass.: MIT Press, 1979.

[9] J. Korein, An Investigation of Reach, Ph.D Thesis Proposal (unpublished), Computer and Information Science Dept., University of Pennsylvania, Aug. 1981.

[10] A. Kumar, Characterization of Manipulator Geometry, Ph.D. Thesis, University of Houston, Feb. 1980.

[11] A. Kumar and K. Waldron, "The Workspaces of a Mechanical Manipulator," Journal of Mechanical Design, Vol. 103, pp. 665-672, July 1981.

---

* There are three degrees of freedom over a surface: translation in two orthogonal directions and rotation.

[12] R. Paul, B. Shimano and G. E. Mayer, "Kinematic Control Equations for Simple Manipulators," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-11, pp. 449-455, June 1981.

[13] D. L. Pieper, The Kinematics of Manipulators under Computer Control, Ph.D. Thesis, Stanford Univ Oct. 1968.

[14] B. Roth, "Robots," Applied Mechanics Reviews, Vol. 31, No. 11, Nov., 1978, pp. 1511-1519

[15] B. Roth, "Kinematic Design for Manipulation," "Workshop on the Research Needed to Advance the State of Knowledge in Robotics," org. by Birk, John and Kelley, Robert, Newport, Rhode Island, pp. 110-118, April 1980.

[16] K. Sugimoto and J. Duffy, "Determination of Extreme Distances of a Robot Hand - Part 1 A General Theory," Journal of Mechanical Design, Vol. 103, pp. 631-636, July 1981.

[17] K. Sugimoto and J. Duffy, "Determination of Extreme Distances of a Robot Hand - Part 2 Robot Arms with Special Geometry," Journal of Mechanical Design, Vol. 103, pp. 776-783, Oct. 1981.

[18] D. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Transactions on Man-Machine Systems, Vol. MMS-10, pp. 7-53, June 1969.

An Architecture for the Design of Large Scale Intelligent Teaching Systems

Gordon McCalla, Darwyn Peachey, Blake Ward

Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, CANADA

Abstract

This paper describes an architecture to support the design of large scale teaching systems, an architecture which is more general and sophisticated than traditional computer assisted instruction (CAI) frameworks and which has been developed with an awareness of the issues raised by artificial intelligence work on CAI. The architecture suggests that course material can be represented at many levels of detail using nested AND/OR prerequisite graphs. This makes it relatively easy to construct a course containing a variety of concepts to be presented in various styles and at various levels of detail appropriate to each concept and each student. When a concept is not understood by a student, appropriate remedial action can be taken without needing to be explicitly pre-planned. Such remedial action is dependent not only on the unlearned concept itself, but also the student's performance so far in the entire course. The framework is designed so as to be easily modified, hence encouraging the course builder to experiment with various versions of the course. In fact, a long-term goal of this research is to provide a highly interactive, flexible course-writing environment for course designers.

1. Introduction

The eventual aim of this research is to produce a very friendly environment for course designers to use in building a course. The environment should allow the course designer to easily construct parts of a course and test them out; to monitor the course parts in action; and to expand and modify these parts as their strengths and weaknesses are understood. Obviously the environment should be interactive. In addition nice graphics capabilities (e.g. allowing multi-colours, providing the ability to window through parts of a course, giving the user a mouse or joy-stick) would enhance the environment substantially. Such a course-writing tool would compare (favourably) to traditional course-writing languages (e.g. TUTOR or NATAL) much as a nice LISP environment compares (favourably) to more traditional programming languages (e.g. FORTRAN, COBOL, or PL/I).

The most crucial requirment of such an environment is the provision of an appropriate set of primitives which represent what a course is and which the course designer uses to conceptualize a course. These must be high level primitives if the designer is to be spared the tedium of explicitly specifying every last bell and whistle in a course. They not only should consist of high level data structures, but also high level control structures in order that the designer can think about the course in as non-procedural a fashion as possible. Finally, they should be general enough to allow a variety of courses to be specified.

There seem to be two basic approaches to the design of these conceptual primitives: (I) a traditional frame-based approach to computer assisted instruction (TCAI) and (II) a more knowledge based approach using some of the principles developed in artificial intelligence work on CAI (ICAI). Lets look more closely at these two approaches.

Traditional approaches to CAI contain at their core an explicit graph suggesting various paths a student may take through a body of course material. Although more recent "generative" approaches to CAI (Chambers and Sprecher (1980)) have led to some flexibility in the way this material is presented and tested, the need to explicitly predict control paths still leads to courses which are quite rigid and hard to design. Moreover, many aspects of a good teaching system tend to be ignored, aspects which many AI-based systems attempt to take into account.

SCHOLAR (Collins and Warnock (1974)), for example, introduces the concept of knowledge representation to a teaching model. The SCHOLAR system has access to a knowledge base of concepts which allows it to in some sense "know" its subject domain of South American geography. The SOPHIE system (Brown, Burton and de Kleer (1981)) also has knowledge of its (circuit design) domain, allowing it to detect bugs in students' solutions to circuit design problems. In BUGGY, Brown and van Lehn (1980) incorporate an entire theory of bugs that students make in solving whole number subtraction problems. Goldstein (1979) has devised a structure called the "genetic graph" not to represent paths through a body of course material, but to represent the various approaches a student takes when learning these concepts. The genetic graph forms the basis for a student model.

All of these AI systems have in common the fact that they severely limit the subject domain in order to manageably study their particular

approach. The problem of how to represent a large body of interacting concepts has been downplayed. Instead, "coaching" techniques for a particular small domain have been developed. It is not obvious how (or even if) these techniques can be extended to larger subject domains, nor is it obvious whether the techniques are well formulated enough to survive the results of further investigation.

So, we are left with the problem of choosing between the traditional and artificial intelligence approaches as our basic paradigm for representing the conceptual primitives of our system. We attempt to steer a middle course between the two approaches. As in traditional CAI, the basic structures being represented revolve about a body of material to be presented (rather than, for example, student modelling or knowledge of the domain), but these structures are considerably more flexible and elegant than those of TCAI. As in the genetic graph, a student model (and student history) is kept, but it is formulated in terms of how well a student is doing on the course material and not in terms of his/her evolving learning structures. Similarly, student errors are diagnosed and the appropriate material which is in error can be pinpointed, but, while this diagnosis is sophisticated, it is by no means a theory of bugs such as BUGGY constitutes for its domain. Hopefully, further developments in ICAI will lead to generally agreed upon and widely applicable theories. This would allow a more useful set of primitives to be incorporated into the course architecture to handle student models, knowledge representation, etc.

In the next section of the paper an overview of the course architecture is given, and this is further elaborated in section III. Section IV discusses our experiences with the construction of a portion of a LISP course in the architecture. Section V sums up what has been accomplished and how it relates to the long term goals of the research.

II. An Overview of the Architecture of a Course

II.1 AND/OR Structure

As in Peachey (1982)[1] course material is represented in an AND/OR course graph (such as the one shown in Figure 1 which displays part of a LISP course). The nodes represent individual concepts; the links between nodes represent prerequisite relationships (not flow of control). An arc connecting links represents an AND relationship among the prerequisites implying that all prerequisites must be learned before the father node can be presented. Otherwise, the prerequisites are ORed, suggesting an alternate order of presentation at the same level of detail. For example, in Figure 1 the prerequisite concepts for lambda-expressions are either the three concepts QUOTE, list manipulation functions, and predicates; or, alternatively, the two concepts multiple argument functions and single argument functions.

The AND/OR structure has exactly the same semantics as in STRIPS (Nilsson (1971)) except that instead of designating a node as "solved", "unsolved", or "unsolvable", it is designated as "learned", "unlearned", or "futile". Initially, all nodes start off as "unlearned" - the task of the automated tutor is to guide the student through the course graph presenting each concept (until it can be deemed "learned") keeping in mind that prerequisites must be satisfied at any stage. Hence, in the LISP course, data structures are presented first; then basic function calling notation; then either multiple argument functions and single argument functions, or QUOTE, list manipulation functions, and predicates; then lambda-expressions; and finally recursion.

Once the last node (recursion in this case) is changed from "unlearned" to "learned", the student is deemed to have learned the entire graph. Occasionally, a concept can be deemed to be unlearnable in which case the corresponding node is marked "futile". Unless there are OR paths through the graph, a futile node can block further progress towards learning the entire graph. Hence, in the same way that it is possible to have unsolvable problems in problem solving, so it is possible to have a futile learning situation in this course architecture.

What advantages does the AND/OR structure offer? The AND structure allows a course designer to very naturally encode the prerequisite relationships of a course without necessarily implying an ordering on the conjuncts. The OR structure allows the designer to specify alternate presentation paths in case certain approaches prove inappropriate for a particular student or simpley to provide variety in presentation style. But, most importantly, the designer needn't worry about flow of control, i.e. which node to present next. The AND/OR rule (that a node is unlearned if any of its AND prerequisites are unlearned or if all of its OR prerequisites are unlearned) can be applied recursively to re-compute these fringe nodes.

II.2 Node Structure

So far, the nature of a node has not been specified. Each node contains the material necessary to present and test the concept represented by the node. The designer has full flexibility in presenting this material any way that suits the material and the student. Thus, material could be presented loosely in a very open-ended learning situation (such as a coaching environment) or more prescriptively using a pre-specified text with standard questions that test the comprehension of the text. The only commitment the architecture imposes on a node is that after presenting a concept to a student, there should be a list

[1] This work expands on concepts developed in Peachey's forthcoming M.Sc. thesis. Much of the basic architecture is his; we extend the thesis work in the area of levels of detail within a node and the interaction of the level of detail hierarchy with the AND/OR structure. The development of the LISP course is also new (Peachey's examples involved teaching data structures and basic economics).

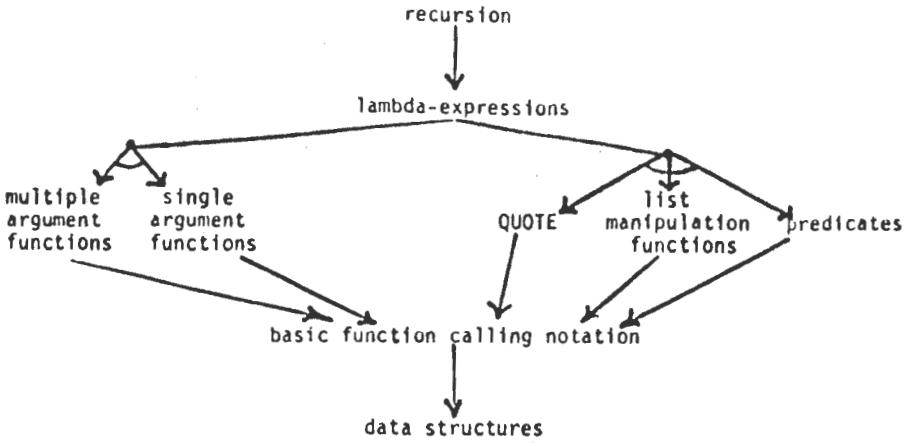


Figure 1 - Basics of LISP

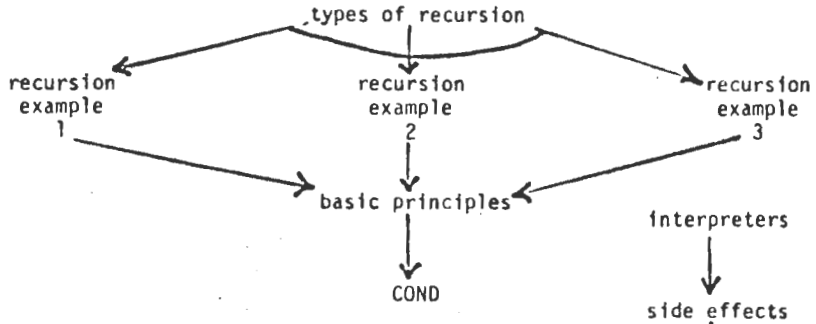


Figure 2 - Recursion

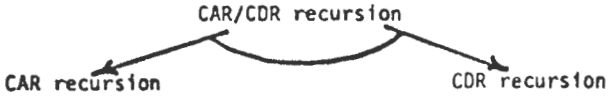


Figure 3 - Types of Recursion

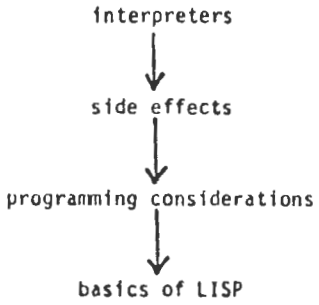


Figure 4 - A LISP Course

of nodes diagnosed as learned as a result of presenting the concept and another list of nodes diagnosed as unlearned. The most common diagnosis is that the node itself is learned or (rarely) futile, although occasionally it is possible to determine that certain previously learned concepts are shaky or certain concepts further on in the course are already well understood. In any event, these lists are used to update the status of nodes in the graph whereupon a new fringe of unlearned nodes can be computed. Note that this approach allows the pace of presentation to be automatically adjusted to a particular student's talents.

Strict node modularity is essential if sophisticated courses are to be created. However, providing nothing more in a node would put a heavy burden on the shoulders of the course designer. So, the course designer is given the option of creating nodes which are not considered merely to be primitive modules, but also can be broken into AND/OR sub-graphs at a lower level of detail. For example, the recursion node in the graph of Figure 1 could be further broken down into the sub-graph of Figure 2, and the types of recursion node in Figure 2 could be broken down into the sub-sub-graph in Figure 3. In fact, Figure 1 itself could be a sub-graph contained in the complete LISP course graph represented in Figure 4. To initiate the presentation of a given node, then, merely involves presenting the earliest unlearned node in the sub-graph which is contained in that node. This rule can be applied recursively down to some primitive "hard wired" level.

II.3 The Interaction Between AND/OR Structure and Node Structure

The interaction of the level of detail hierarchy with the AND/OR hierarchy is an important issue. A further rule is introduced to handle this - a node which has been "opened" is unlearned if the last node (the top-most node in these examples) of its sub-graph is unlearned. Using this rule plus the usual AND/OR rules it is possible to determine not only a fringe of unlearned nodes, but also a level at which to present the nodes. Normally this level will be the most primitive level (since all nodes start off as unlearned), but the designer has the option of specifying the last node of a sub-graph as learned and hence can prevent any given node from being opened. This gives the designer the ability to present various parts of the course at appropriate levels of abstraction. It also allows a concept that is being reviewed to be presented at different levels of detail than it was the first time through (this is possible because previous nodes at any level of detail can be diagnosed as unlearned).

So, by appropriately specifying nodes as learned or unlearned, the course designer can set up fast or slow, detailed or abstract courses for a particular student. Alternate styles of presentation at the same level of detail can even be ensured by placing futile indicators on nodes along certain OR paths forcing presentation of other paths. These indicators are kept in an instantiated version of the course graph which constitutes a student model. The student model is continuously updated as the student goes through a course, thus providing an ongoing model of the state of the student's mastery of the course material. In addition a student history is maintained to keep a detailed outline of the actual sequence of nodes presented and some details on student behaviour at each node. The student history is used, among other things, to provide variety when reviewing the concepts of previously presented nodes - OR paths unused on the earlier pass are taken the next time through if possible.

III. Details of the Course Presentation Methodology

III.1 Selecting the Node to Present

Assume that a particular sub-graph has been chosen at the appropriate level of detail. AND/OR semantics will provide a fringe of unlearned nodes in the graph. The first problem is to select one such node to present to the student.

Any unlearned fringe node with a futile node on the path between it and the goal (last) node can be eliminated from consideration since it will be impossible to get to the goal node in any event. To further refine the choice it is important to realize that there are two kinds of unlearned nodes: those that have never been presented and those that have been presented but have been diagnosed as unlearned after the presentation. In order to increase the variety of nodes presented, all unlearned but previously presented nodes are rejected (unless no others are available). This ensures, for example, that alternative paths will be chosen over previously unsuccessful paths. It also ensures that, even in the case of conjunctive subgoals, the other conjunctive nodes will be chosen before returning to the unsuccessful nodes. This may allow the student to achieve a deeper understanding of the subject before once again trying something he or she had trouble with before.

If there are still many choices, perhaps they can be distinguished by how far they are from the goal node. Nodes furthest from the goal can be rejected. Finally, to choose between any equidistant nodes, indicators associated with the nodes can be examined. Each node has two kinds of indicators: an importance marker, indicating how crucial a node is as a prerequisite to another node; and a criticality marker indicating how crucial a node is to the supernode in which it is contained. In this context, only the importance markers matter - the most important prerequisite will be chosen. If the nodes still aren't distinguishable, a random choice is made.

Once a fringe node is chosen, another decision must be made: whether or not to open the node.

III.2 Deciding Whether or Not to Open a Node

Deciding whether or not to open a node is a non-trivial task. Obviously, primitive nodes are not opened, but are just presented as is. Non-primitive nodes are opened or not using a procedure that takes into account the following two factors. First, the node itself contains a recommendation as to whether or not it should be opened. Second, sub-nodes of the node are examined in a more subtle way than indicated in II.3. The existence of lots of unlearned nodes of high criticality suggests the node should be opened in order to present these concepts. The lack of such nodes suggests the node should not be opened, either because there are not many sub-nodes of high criticality or they are already learned. These factors are combined in a rather arbitrary fashion at present, a method which will likely be subject to considerable change as experimentation with the system continues. Basically, an evaluation function is used to combine the two factors. In this function the first factor dominates if most of the sub-nodes are unlearned and unpresented. Otherwise the second factor (the sub-node statuses) dominates the decision.

Once a node is to be opened, two problems arise: first, which node in the sub-graph to choose (already discussed in III.1); and second, whether or not to present some or all of the material at the current level of detail and if so whether to do it before or after the sub-graph is presented.

III.3 Node Presentation Styles

In addition to possibly having a sub-graph, a node can contain "primitive" material to be directly presented to the student, material that in a sense duplicates the content of the sub-graph but at a higher level of detail. For the purposes of this discussion it is convenient to consider that this material consists of two components: a presentation part (called the "blurb") and a testing part. Of course, the material in many nodes will not have this structure (e.g. testing and presentation will be mixed or the node will provide a coaching environment), in which case the discussion is less applicable although some lessons can still be drawn depending on the exact structure of the node.

Given that a node is not opened, there are three possible presentation styles to consider: (i) just give the blurb (i.e. decide not to test, perhaps because this is not an important node or is just being reviewed); (ii) give both the blurb and the test; or (iii) just test the concepts represented by the node but don't give the blurb (perhaps because the node has been presented before but has later been deemed unlearned).

If a node is to be opened, there are four possible styles: (i) give the blurb, then expand the node, and test the node on the way back out; (ii) give the blurb, then expand the node, but don't test it at this level at all; (iii) expand the node and test it on the way back out, but don't give the blurb at this level (presumably because the sub-nodes have provided enough verbiage); and (iv) just expand the node, but don't give the blurb or test it at this level (presumably because the sub-nodes have done these jobs). Styles (i) and (ii) are top-down styles; style (iii) is bottom-up; and style (iv) makes no concessions to higher levels of detail at all. Choosing among these is not easy, although style (ii) seems most widely appropriate because of its ability to give successively refined overviews of the material without the tedium of multiple levels of testing. We hope that further experimentation will provide more insights on which styles to choose under which circumstances.

III.4 After Presentation of a Node

After a node has been presented, a diagnosis must be made as to what has been learned or not. This is usually done by the testing component (if such exists in the node) on the basis of student performance on the test. However, a more qualitative diagnosis process can be imagined, based on "over the shoulder" observations of the student as he/she "plays" with concepts presented in the node.

When the node decides that a certain concept, represented by some other node, is not well understood, it must come up with a number that indicates the severity of the problem. Similarly, when the node decides (more rarely) that some other node is, in fact, already well understood, it must come up with a number indicating how well comprehended the concept represented by that node is. Such numbers are used to appropriately decrease (or increase) a "learned threshold" associated with the poorly (or well) understood nodes. If this threshold drops below a certain number, the node is diagnosed as unlearned; if it rises above a certain number, the node is diagnosed as learned (even if it is unpresented as yet!); and if it drops off the bottom of the scale, the node is diagnosed as futile. Note that it is not necessary to reach a full diagnosis of some node all at once. For example, a previously learned node can have its threshold decreased without yet making it unlearned. Future diagnosis could continue to decrease confidence in the student's understanding of that node's expertise until it finally slipped into unlearned status. In this way diagnoses can be accumulated over time.

Minimally, the current node should keep control until it has reached a definitive diagnosis of itself (if not of any other nodes). Once such a diagnosis is made, the node is finished and the learned, unlearned, and futile markers can be propagated throughout the entire course graph.

III.5 Computing the New Fringe

The process of computing a new fringe of unlearned nodes proceeds using basic AND/OR semantics. In essence any node with unlearned or futile prerequisites is itself unlearned or futile unless there is an OR path without such unlearned or futile prerequisites. In actuality the process is more subtle than this. Importance markers are taken into account in deciding whether to propagate an unlearned or futile status - any relatively unimportant prerequisites may not have sufficient weight to count.

Once the unlearned/futile propagation is done, it must be decided whether the whole sub-graph is learned (or futile). As mentioned in II.3 if the last (goal) node of the sub-graph is learned (or futile), then so is the whole sub-graph. Once again, though, the whole process is more subtle than this. Basically, a sub-graph can also be deemed to be learned (or futile) if enough of its high criticality nodes are learned (or futile), under the assumption that the other nodes aren't significant contributors to the sub-graph status. When a learned (or futile) status is given to the entire sub-graph, then it is also given to the super-node containing the sub-graph, and the whole propagation process can be repeated at the higher level and so on recursively. Note that the upwards propagation described here when combined with the decision as to whether or not to expand a node determines automatically a level of detail at which to present portions of the course.

But, under the assumption that the upwards propagation does not occur, a fringe must be computed at the current level. A node is on the fringe if there are nothing but unlearned nodes on at least one path connecting it to the goal node and if just previous to it (on all OR paths to the source (first) node) there are nothing but learned nodes. It is from the fringe nodes that a new node to present can be chosen (see III.1).

IV. The LISP Course

The entire architecture described above has been implemented and we have begun the construction of a LISP course using this architecture. Figures 1 - 4 give a flavour if the kinds of graphs present in the course. Most of the "basics" node (Figure 4) has been implemented at several levels of detail and a preliminary version tested on a couple of students. Unfortunately, bugs in the program prevented a serious evaluation of the usability of the course. However, the course itself was built in about 60 hours of (on and off) work over the busy fall term. Given that some debugging of the course architecture itself was required as the LISP course was built, this augers well for the usefulness of the architecture to the course designer.

The LISP course, as designed so far, is not perfect. Many of the nodes are too wordy and lockstep the student into a relatively boring sequence of fill-in-the-blank, multiple choice, or yes/no questions (boring despite the fact that the questions generated vary each time through a node, thus at least providing some variety). But the nodes which test functions (e.g. lambda expressions in Figure 1) are not quite so mundane. The student is asked to write simple lambda expressions to achieve list twiddling examples. We have implemented a "smart" EVAL to execute lambda expressions. It does a syntactic analysis of the lambda expression, steps through the lambda expression explaining to the student what is happening at each stage and allowing her/him to stop and look around at each stage. A few standard bugs are recognized, as well, in order to aid diagnosis.

This is just the first step in making the whole course more sophisticated. For example, ideas such as those in Fine (1977) will need to be incorporated into the system to make it more knowledgeable. Whether (or more likely how) to augment our architecture with such things as semantic networks, pattern directed invocation schemes, etc. will have to await more extensive testing of the current architecture and course.

V. Conclusion

In this paper we have discussed an architecture for the design of fairly sophisticated courses. The emphasis has been on providing controlled flexibility to the course designer. The control is enforced through the rules for AND/OR prerequisite structure as extended to many levels of detail. The flexibility is provided through the modularity of nodes which gives the course designer full scope at any node, notwithstanding the minimal interdependencies imposed by the above rules. Such an architecture seems to provide the foundation for a nice course writing environment. The AND/OR and level of detail structure form the basis for powerful primitives which the course designer can employ directly; the modularity allows the course designer to incrementally refine and extend a course without undue upheaval.

It must be emphasized, however, that the architecture is just the beginning of a nice course writing environment. It needs to be extended to handle other facets of the teaching/learning process - e.g. more sophisticated models of how students learn, more sophisticated knowledge representation techniques to allow the system to do more self-analysis and make better inferences, etc. It needs to be embedded into a better interactive environment where there are graphics that allow sub-graphs to be displayed in separate windows and open (revealed) or closed (hidden) at will; with facilities that allow the student (or course designer) to see the nodes being expanded, markers being propagated, etc.; with system provided primitives for node creation, destruction, and manipulation. A commitment to this architecture suggest that this last, at least, should be relatively easy and that the primitives can be very powerful when compared to most course-writing tools.

More esoterically, the course designer will need trace facilities, pre-packaged student models representing various stereotypical students, and perhaps, in order to test out a course, even active procedures (sort of simulated students) which could actually "take" the course. The teacher who is monitoring a student will need to be provided with better tools as well. These could include the ability of the teacher to be hooked to a "slave" terminal and from there to be able to smoothly interrupt or direct both the student and the course (e.g. to override the course in the choice of material to present next). All of this would be, of course, a huge amount of work and would require the solution to a number of very hard problems. Nevertheless we feel this is an interesting start.

Acknowledgements

We would like to acknowledge the National Sciences and Engineering Research Council of Canada and the University of Saskatchewan for providing grant money for this research from the President's NSERC fund.

References

Brown, J.S., Burton, R.R., and de Kleer, J. "Pedagogical, Natural Language, and Knowledge Engineering Techniques in SOPHIE I, II, and III" in Intelligent Tutoring Systems, Sleeman, D., and Brown, J.S. (eds.), Academic Press, 1981.

Brown, J.S. and van Lehn, K. "Towards a Generative Theory of Bugs", Xerox PARC Technical Note, 1980.

Chambers, J. and Sprecher, J. "Computer Assisted Instruction: Current Trends and Critical Issues", C.ACM 23, 6, June 1980, pp. 332-347.

Collins, A.M. and Warnock, E.H. Semantic Networks, Tech. Report #2833, BBN, Cambridge, Mass., 1974.

Fine, G. Design of an Intelligent LISP CAI Tutor, M.Sc. Thesis, UBC, Vancouver, 1977.

Goldstein, I. "The Genetic Graph: A Representation for the Evolution of Procedural Knowledge", Int. J. of Man-Machine Studies 11. 1. January 1979, pp. 51-77.

Nilsson, N.J. Problem Solving Methods in Artifical Intelligence, McGraw-Hill, 1971.

Peachey, D. Course Graph Structuring and Diagnosis of Learning Problems in an Intelligent Teaching System, M.Sc. Thesis, Dept. of Computational Science, U. of Saskatchewan, forthcoming, 1982.

An Expert Consultant for the UNIX Operating System: Bridging the Gap Between the User and Command Language Semantics

Robert J. Douglass
Los Alamos National Laboratory
Los Alamos, NM 87545

Stephen J. Hegner
University of Virginia
Charlottesville, Va. 22901

1. INTRODUCTION

The UNIX Computer Consultant (UCC) is an expert system currently under development at the Los Alamos National Laboratory and at the University of Virginia. It is designed to aid both novice and advanced users of the UNIX operating system in their use of the system.

UCC is far more powerful than the typical operating system help utility, which provides on-line retrieval of only documentation of specifically named commands. Thus, a user who does not know the name of the command he needs, or who needs information on items other than command-specific definitions (such as information on the organization of the operating system's file system) can get little or no help from such a help utility. UCC, on the other hand, contains a sophisticated retrieval facility that can key on a number of concepts, including not only specific command definitions but also logical operations without a given command (e.g., listing a text file) and concept definitions (e.g., a UNIX path). Furthermore, UCC, unlike existing help systems, can respond with a specific answer given a specific question.

Because UCC is designed to be a consultant for an existing system, it is not integrated into specific utilities, but rather exists as an independent subsystem on top of the operating system. This is in contrast to some systems that incorporate user support facilities within the utility. While the direct integration approach shows much promise for help facilities for newly designed systems, its applicability to existing systems (such as UNIX) is made difficult by the necessity of rewriting part of the existing system, in addition to the entire help facility. Furthermore, such direct incorporation of help features make the answering of queries not keyed on specific commands very difficult.

1.1. An Overview of UCC

The UCC system consists of two major modules, the front-end module and the UNIX knowledge base and solver module. The major role of the front end is to translate a user's natural language query into the specially designed formal query language UCCquel and to translate answers to UCCquel queries back into natural language. The role of the UNIX knowledge base and solver module is to produce answers to the formal UCCquel queries.

The front-end module accepts questions from users such as "How do I list a directory file?" and calls a lexical analyzer to delete noise words from the question, convert synonyms and tensed words to a standard token, and identify unknown words. The tokenized question is then parsed by an augmented transition network (ATN), phrase by phrase, if an attempt to parse an entire sentence fails. The ATN produces three pieces of information: a) the type of question; b) a template, called a case frame, describing the main operating system action mentioned in the question; and c) a set of predicates, derived from noun phrases, prepositional phrases, adverbs, and other modifying phrases and clauses that assert properties of entities referred to in the question (such as files).

The parser passes its output to a query generator which produces a) a formal query with a set of predicates describing a function to be performed by the operating system; b) preconditions describing the input to that function; and c) postconditions describing the function's output. The formal query will contain unbound variables that must be instantiated by the knowledge base and solver module to answer the original question.

The knowledge base and solver module contains descriptions of UNIX commands and concepts, and a set of propositional semantic definitions of commands similar to Hoare semantics [1]. The descriptions are used to answer questions such as "What is a directory file?" or "What does 'ls' do?" The formal semantic definitions are used to answer questions such as "How do I list a directory file?"

Once the variables in a query have been bound by the knowledge base, then the instantiated query is passed back to the front end to produce an answer in English for the user. The front end formulates an answer by noting what type of question was asked, selecting an appropriate template for an answer, and using a dictionary of predicate definitions to describe the relevant parts of the instantiated query.

In this paper, we discuss briefly the nature of the formal query language UCCquel, and then we detail the process of transforming a natural-language query into UCCquel. In a future paper, we will detail the workings of the UNIX knowledge base and solver module.

2. QUERY MODELING AND CLASSIFICATION

To understand the operation of the natural-language front end, it is first necessary to understand the formal queries that it is to produce. In this section, we outline the nature of the various queries that are considered and show how they are represented in UCCquel.

2.1. Static Queries

Static queries are the most basic type that are handled by UCC. A static query is one that requests information not involving system dynamics, and, expressed in natural language, are typically of the "What is ... ?" variety. Examples include "What is a pipe?" and "What is a home directory?" In each case, a simple retrieval of a

definition is all that is necessary. In essence, a static query is just a retrieval of the form "get {x | P(x)}" (which is the form of a nonprocedural query to a relational database [2]). For example, "What is a pipe?" translates into the formal query "get {x | x is the definition of a pipe }.

All queries in UCCquel are expressed in an LISP-like notation. Our representation of this query looks like

```
(Find X
   (Basic-object-description ("pipe") = X)).
```

X is the free variable that must be bound in answering the query.

2.2. Dynamic Queries

To be useful, UCC must also have the capability of processing dynamic queries. These queries differ from their static counterparts in that they deal with the dynamic aspects of the UNIX operating system. Examples include "How do I list the contents of a file paged and with page headers?" and "What happens if I try to print a directory file with the pr command?" In each case, the query deals with the determination or action of a system command.

We may still classify such queries as "get {x|P(x)}," provided we choose P appropriately. In the case of a static query, P is just a well-formed formula (wff) in the (ordinary) logic that describes the retrieval of concept definitions. In the case of dynamic queries, P must be replaced by a wff in an appropriate dynamic logic because we are now dealing with system dynamics. For the purposes of UCC, the usual propositional-style semantics, commonly used in programming language semantic definition [1], provides a convenient framework. A wff in this style of semantics takes the general form {R} F {Q}, where R and Q are wffs in an appropriate static first order logic (known as the underlying base logic), and F is an action that may change the truth values of statements in the base logic.

As a particular example, consider the query given above "How do I list the contents of a file paged and with page headers?" Here R is of the form "#x is a (generic) text file" and Q is of the form "The contents of the standard listing device is what the contents of the file #x was before the command, modified to be paged with standard page headers." F is an unknown operator, to be found in answering the query. In the retrieval, P(x) is {R} X {Q}.

2.3. Fundamental Types of Dynamic Queries

A dynamic query of the form "get { x | P(x) }," where P = {R} F {Q}, may be placed into one of eight possible classifications, depending upon which subset of R, F, and Q is known and which is not. In the initial implementation of UCC, only two of these will be considered.

(1) R and Q are known, but F is not. This is the general structure of a "How do I ..." type of query and is illustrated by the example in section 2.2.

(2) R and F are known, but Q is not. This is the general structure of a "What happens if ... ?" type of query. As an example, once again consider "What happens if I try to print a directory file using the 'pr' command?" Here R is "#x is a (generic) directory file" and F is "pr #x," with Q to be found.

2.4. Secondary Responses in Dynamic Queries

While the classification of dynamic queries as outlined in the previous section is a useful guideline, it is not always completely adequate. For example, consider once again the query "How do I list the contents of a file paged and with page headers?" To answer this query with R and Q as given in section 2.2, it is not sufficient to simply supply the response "pr #x," because, for this response to be correct, we must have that #x is a file that is currently readable by the user. In responding with the answer, we must add this readability condition to the preconditions R. We term such an addition a secondary response. In the formalization of dynamic queries in UCCquel, we always permit the addition of appropriate secondary responses.

2.5. An Example UCCquel Query

Below is displayed the UCCquel query for "How do I list the contents of a file paged and with page headers?"

```
(Find (R1 F1 Q1)
  (Dyn R F Q)
    (Define R
      ((Clauses
         (file #x (type = "text")))
       (addnecessary R1)))
    (Define F
      (( transform F1)
       (logical (list-text-file))))
    (Define Q
      ((Clauses
         (contents(%user-terminal)
           = modified (contents (time (-1,file(#x)))
                 (type = "text")
                 (paged = "yes")
                 (pageheaders = "yes")
                 (value standard))))
       (addimplications Q1))))
```

P1, F1, and Q1 are the variables to be bound in the solution of the query. (Dyn R F Q) means {R} F {Q}. F1 is the fundamental variable to be bound and is declared to be a transform. P1 and Q1 are to be bound to secondary preconditions and postconditions, respectively. The definitions of R and Q are wffs in our underlying logic known as the static definition logic. The "time(-1 ...)" notation in the postcondition Q is used to ensure that the value of the user terminal after the execution of the command will be the contents of #x before the command. This is merely a shorthand and does not violate the constraint that R and Q be statements in a static logic; by setting #x to a dummy file @x in the preconditions and using file @x instead of time(-1,file (#x)), we can stay entirely within the constraints. The "logical ... " part of the definition of F is indicating which case frame was used in constructing the query, and will be mentioned again in the next section.

2.6. Specific Queries

In the example query illustrated above, the file to be listed is not specified, and so is represented as a generic text file, #x. If the user instead had asked "How do I list the contents of my file /bin/paper paged and with page headers?", the formulation would differ fundamentally in that #x would be replaced by /bin/paper. Also, the assumption that /bin/paper is a text file must be dropped. Rather, it must be left to the knowledge base to decide whether or not /bin/paper is listable.

3. THE INTERFACE BETWEEN THE USER AND THE KNOWLEDGE BASE

The front-end module is responsible for converting a user's English questions into formal UCCquel queries and formulating English responses. This process consists of five levels of analysis: lexical (or word), phrase, clause, sentence, paragraph (or dialogue), and topic. Five major data structures are used by these levels, including a dictionary, a set of case frames, context registers, predicate descriptions, and answer frames. We will examine these levels and data structures more closely as we follow the front-end processing of the question "How do I list a file paged and with page headers?" This question is the input to the front-end module and the output is the UCCquel query given in section 2.5. Only the analysis of a user's question will be discussed here; we leave a discussion of the more straightforward process of generating an answer to a future paper.

3.1. Lexical Analysis

Lexical or word-level analysis is performed by a tokenizing routine. Each time it is called by the parser, it scans the English question and returns the next token. The tokenizer looks up words in the dictionary to see if they are defined and replaces them with standard tokens if they are found. It catches multiword idioms and common noun-noun modification, such as "how do I" or "directory file"; it also flags unknown words, identifies file names when possible, replaces synonyms with a standard root, and replaces tensed and pluralized words with a standard token plus the features indicating tense or number.

The tokenizer uses a token definition stored in the dictionary and a look-ahead mechanism to determine what English words or UNIX file names should be grouped into one token. Words with no dictionary entry are passed along to the ATN because they may be names of specific files.

For example, given our sample question "How do I list ...," the tokenizer would return the following tokens (the $ designates a token): $hwd (representing the phrase "how do I"), $i (for "I"), $list, $a, $file, $with, $phead (plural) (for page headers), $and, and $paged.

3.2. Phrase-Level Analysis

Tokenized questions are parsed, phrase by phrase, using a grammar represented by an augmented transition network (ATN). ATNs are a standard tool for parsing natural language [2]. Although other parsing techniques have proved useful for parsing restricted English questions [3,4], ATNs can parse statements that include ellipsis and grammatical errors.

UCC's ATN produces a parse tree for a question, either for a whole sentence or phrase-by-phrase. It uncovers and enforces syntactic rules and semantic constraints within a phrase. Predicates are generated by the ATN to represent the meaning of the phrases. These predicates form the preconditions, {R}, and postconditions, {Q}, for dynamic queries.

The ATNs also select a small set of case frame candidates that could correspond to the main action described in a clause. Case frames and their role are described in the next section.

A parse of our sample question would produce the following results.

(1) Question Type: hwd (meaning "how do I").

(2) Phrase-Level Semantics:

Noun groups: (NG1 (file #x) (type = "text")
(paged = "yes"))
(NG2 (pageheaders))

Verb groups: (VG1 (verb list) (direct-object NG1)
(tense present))

Preposition groups: (PG1 (with NG2))

(3) Case Frames:

(logical list-text-file)
(logical list-directory)
(logical enumerate)

3.3. Clause-Level Analysis

The ATN parses the phrases in a question to produce predicates describing the noun groups and preposition groups. These phrases are grouped at the clause level by a "case-frame-fitter." Case frames are templates representing the main action of a clause and the constituents of the action, such as the actor and receipient of the action; usually, they correspond to the main verb in the clause.

Several natural-language understanding systems have used case frames to represent the action in a sentence [2,5]. For Schank and his coworkers, a small set of conceptual case frames represent all actions expressed in natural language [5]. In UCC, case frames correspond to logical operations in an operating system, and they form the main link between English language operating system concepts and the formal semantic definitions of specific UNIX commands. Some have a direct correspondence to UNIX commands, while others may be associated with several different commands that could be used to accomplish the same logical operation.

For example, as shown in the last section, the verb "list" could represent one of three case frames, list-text-file, list-directory, or enumeration. List-text-file is associated in the knowledge base with three UNIX commands for listing a text file, "cat," "pr," and "more." The case-frame-fitter must select one of the three possible case frames using the parsed question and semantic constraints specified in the case frame, and the knowledge base and solver module must decide which UNIX operator associated with the list-text-file case frame, "cat," "pr," or "more," is the appropriate answer to the user's question.

Case frames contain default information on preconditions and postconditions associated with the logical action represented by the case frame. They also specify slots to be filled from a user's question or from context, and they provide semantic constraints on what information can fill those slots.

In our example, the case-frame-fitter selects the list-text-file case frame because the direct object of "list," identified by the ATN as the noun group "a file," is assumed to be a text file. The list-directory case frame would have been selected if the object of "list" had been a directory, and the enumeration case frame would have been picked if the object had been a noun group that could be enumerated, such as "the number of users currently logged on the system."

Once the list-text-file case frame is chosen, the case-frame-fitter notes that the list-text-file specifies that the question should contain phrases describing how the text is listed. In our example, the case-frame-fitter finds the phrases "paged and with page headers" and fits these into slots that modify the description of the result of the logical operation list-text-file. This description

constitutes the postconditions, {Q}. For our example, the preconditions will be the direct object of "list" or the phrase "a file." The case frame specifies that the preconditions must be "contents (file (#x)(type = "text"))," and this is consistent with the ATN's parse of the noun group, "a file." Since "a file" is parsed as an indefinite noun group, the case-frame-fitter binds the preconditions to be a generic file, #x.

3.4. Sentence-Level Analysis

Once a case frame has been selected for each clause in a question and the case-frame-fitter has formed {R} and {Q} from the parsed phrases, then the "query-generator" is called to produce a formal UCCquel query. For simple questions (questions stated in one clause), the formal query is essentially built by the case-frame-fitter, and the query-generator has to only verify it with the user and pass it along to the knowledge base and solver module. The formal query generated for our example was given in section 2.5.

The query-generator must integrate several clauses into one formal query in the case of questions that are stated in several clauses (such as "If I ..., then ... ?") and questions stated in several sentences (such as "I have a directory file that I made read only. Why can't I list it?"). Such questions will produce several instantiated case frames, and the query-generator must build one query out of them. Usually, multiple clauses serve the function of further describing the preconditions of the action in the main clause.

3.5. Dialogue and Paragraph-Level Analysis

Work with natural-language database systems and expert consultants [2,6,7] has shown that users usually ask several related questions and that the system must be able to maintain a dialogue with the user and use intersentence (or paragraph level) context. UCC saves contextual information using "context registers" similar to those used by Waltz [2] and by using clarification dialogue.

Context registers are push-down stacks with a short fixed length. They hold information derived from previous questions and answers, such as the last case frames referenced, the pre- and post-conditions of previous questions, and the subjects and objects of previous questions. Ellipsis (missing portions of a question) and pronouns are common in questions, and UCC must use the case frames to fill in this information. If the case-frame-fitter or query-generator finds that a portion of a case frame or query is missing from a question, then they consult the context registers to find the most recent piece of information that will satisfy the semantic constraints of the current question. For example, if the user asks the question "How do I list a file?" and then asks "With page headers?", UCC must resolve the ellipsis in the second question by popping off the most recently used case frame and preconditions from the appropriate context registers.

UCC maintains a dialogue with the user by answering his questions, saving the previous questions in context registers, and using clarification dialogue to verify that a user's question has been correctly understood. In our example, once the query-generator has formed the query, it would verify the query with the user with the following dialogue.

```
UCC: Your query is understood as:
     Find a command that accepts as input.
      1a. a text file.
     and the command will produce output.
      2a. at your terminal
      2b. has page headers
      2c. is paged

    Is this all right? (yes/no):
USER: no
```

Because the user responded "no," UCC will reply with the following

```
UCC:  Do you wish to:
      1. rephrase the question entirely?
      2. change an item?
     Enter 1, 2:
USER: 2
UCC:  Which item(s), 1a, 2a, 2b, or 2c:
USER: 2a
UCC:  What is the new condition?
USER: output goes to the spinwriter
UCC:  Any other changes?
USER: no
UCC:  Your new query is understood as ...
```

Once UCC has verified a query, it is shipped off to the knowledge base and solver module to produce an answer.

3.6. The Topic Level

Users usually ask several questions about a particular UNIX topic, such as questions about using the editor, manipulating files, or compiling a program. To successfully parse and answer questions about a variety of topics, the knowledge base, vocabulary, and case frames have been partitioned into different topic areas. At present, UCC contains information about only the file system and the command language. We are experimenting with a menu-driven system for switching between topics. Complex natural-language processing and question answering require that questions have a limited and well-defined context and that the case frames, vocabulary, and knowledge base be partitioned by topics.

4. CONCLUSION

Natural-language front ends for database systems have been in existence for several years as have expert consultant systems. UCC combines both of these lines of research into a single system that goes well beyond typical operating system help facilities to provide an expert consultant with sophisticated natural-language understanding ability.

We have outlined the process of producing a formal query from a natural-language question and shown how queries can be formally modeled in the UCCquel query language. UCC successfully bridges the gap between users' English questions and formal command definitions by dividing the task into levels of analysis from lexical to topic, and by partitioning the major data structures of UCC by topic. In this way UCC achieves generality and sophisicated language understanding over a broad range of topics. It is intended that UCC serves as a model for the design of expert consultants for other operating systems and for systems that, like operating systems, consist of collections of processes.

5. ACKNOWLEDGMENTS

The authors would like to acknowledge the contribution of David Stotts, Andrew Lacy, Kelton Flinn, and John Taylor to the development and implementation of UCC.

6. REFERENCES

[1] Alagic, S. and M. Arbib, *The Design of Well-Structured and Correct Programs*. Springer-Verlag, 1978.

[2] Waltz, D., "An English Language Question Answering System for a Large Relational Database," *CACM, 21, 7*, 1978, pp. 526-539.

[3] Harris, L., "User Oriented Data Base Query with the Robot Natural Language Query System," *International Journal of Man-Machine Studies, 9*, 1977.

[4] Hendrix, G., E. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data," *ACM Transactions on Database Systems*, 1978.

[5] Schank, R., "Identification of Conceptualizations Underlying Natural Language," *Computer Models of Thought and Language*. R. Schank and K. Colby, eds., W. H. Freedman and Company, 1973, pp. 187-248.

[6] Shortliffe, E., *Computer-Based Medical Consultations: MYCIN*. American Elsevier, New York, 1976.

[7] Michie, D. (ed.), *Expert Systems in the Micro Electronic Age*. Edinburgh University Press, Edinburgh, 1979. 6.

Inference, Incompatible Predicates and Colours

Mary Angela Papalaskaris Lenhart K. Schubert

Computing Science Department
University of Alberta

Abstract

As part of the inference system of a semantic net, we are building a resolution-based theorem prover capable of directly "resolving" not only complementary pairs of literals such as [elephant(Clyde), ¬elephant(x)], but also incompatible pairs such as
[elephant(Clyde), ¬animal(x)],
[elephant(Clyde), canary(x)],
[yellow(Clyde), grey(x)],
[(sort-of elephant)(Clyde), ¬(sort-of animal)(x)],
[(sort-of tan)(Clyde), ¬(sort-of brown)(x)],
[(sort-of yellow)(Clyde), (sort-of blue)(x)].
We had originally expected to handle all such examples of "extended resolution" by means of previously designed efficient algorithms for type lattices. However, we have found that certain classes of "colour resolutions" (exemplified by the last pair of literals above) do not lend themselves to lattice methods. Instead, we have had to augment our representation with another special group of algorithms based on a three dimensional (hue, purity and dilution) colour space.

I. Introduction

A semantic net system in which knowledge is topically organized around concepts has been under development at the University of Alberta for some years (Schubert, Goebel & Cercone 1979, Covington & Schubert 1980). The system is capable of automatic topical classification and insertion of modal logic input sentences, concept and topic oriented retrieval, and property inheritance of a rather general sort.

Efforts are currently under way to extend the inference capabilities of the system, to enable it to answer some of the kinds of questions which people can answer "without thinking". One type of inference which has been under study is efficient inference of inclusion and disjointness relationships in quasi-hierarchies of parts or concepts (types). Specialized data structures and algorithms have been designed for this purpose (Schubert 1979, Papalaskaris & Schubert 1981). We view such special-purpose mechanisms as essential adjuncts to any general inference system based on symbolic logic.

Consider a system whose deductive component is resolution-based (as ours will be – this happens to be natural since the topical classification and retrieval mechanism already requires propositions to be in modal clause form). The following are some trivial deductive problems the deductive component may be faced with, either in answering user questions or in checking new information for inconsistency and redundancy:

1. *Given knowledge:* elephant(Clyde),
   ¬elephant(x) V grey(x)
   Question: ?grey(Clyde)

2. *Given knowledge:* elephant(Clyde),
   plus knowledge about types of animals
   Question: ?animal(Clyde)

3. *Given knowledge:* elephant(Clyde),
   plus knowledge about types of animals
   Question: ?canary(Clyde)

4. *Given knowledge:* yellow(Clyde),
   ¬elephant(x) V grey(x),
   plus knowledge about colours
   Question: ?elephant(Clyde)

Question 1 can be answered by resolving the two complementary "elephant" literals, with result grey(Clyde). In a refutation proof, this would in turn be resolved against the denial ¬grey(Clyde) of the question. The resultant empty clause justifies a "yes" answer.

Question 2 could be answered by a series of resolution steps that progress along the superconcept sequence connecting "elephant" and "animal"; but this is just where we would like instead to invoke special inference methods for type lattices. From the theorem prover's point of view, this should be a one-step inference: in terms of a refutation proof, elephant(Clyde) is incompatible with the denial ¬animal(Clyde) of the question in much the same way that complementary literals are incompatible, and should yield the null "resolvent". This idea can of course be implemented by recognizing "elephant" and "animal" as elements of a type lattice for which special algorithms are available (e.g., McSkimin & Minker 1979, Schubert 1979, Papalaskaris & Schubert 1981). Similarly, it should be possible to obtain a one step disproof for question 3 by "resolving" the incompatible literals elephant(Clyde) and canary(Clyde). In question 4, one "resolving" step should recognize the incompatibility of yellow(Clyde) and grey(x) and hence infer the "resolvent" ¬elephant(Clyde), which then resolves in the proper sense with the question clause, to yield a negative answer.

One interesting question which arises about this sort of "resolving" is whether it can be extended to deal with *modified predicates* such as "large animal", "dark brown" and "sort of brown" (or brown/*sh*). Natural language, after all, provides a large repertoire of predicate modifiers, and presumably any adequate knowledge representation language must contain the logical counterparts of at least some of these.

II. Predicates modified by hedges

We have concentrated our efforts on a particularly troublesome modifier, namely "sort of". It is characteristic of this modifier (and of "hedges" in general) that (sort-of P)(x) fails to entail P(x); this is in contrast with cases like (large P)(x), (typical P)(x), and (dark P)(x). Note however that we can take P(x) to entail (sort-of P)(x). For example, an elephant is

certainly *sort of* an elephant, although the maxims of cooperative conversation (specifically the quantity and brevity maxims) imply that use of the hedge is improper and therefore misleading if the unhedged predicate is known to apply (Grice 1975).

Let us reconsider the (extended) resolution steps postulated in examples (1) – (4) with some of the predicates modified by sort-of. In example (1) the standard resolution [elephant(Clyde), ¬elephant(x)] was required. According to the assumed properties of sort-of, the pair [(sort-of elephant)(Clyde), ¬elephant(x)] is compatible while the pair [elephant(Clyde), ¬(sort-of elephant)(x)] is not. The latter incompatibility is easily detected in two resolution steps given the axiom schema

$$\neg P(x) \vee (\text{sort-of } P)(x),$$

which captures the entailment postulated above. No methods other than standard resolution (in conjunction with rules for applying schemata involving predicate modifiers) appear to be required in this case.

Example (2) called for "resolving" the pair [elephant(Clyde), ¬animal(Clyde)] by special lattice methods, so as to avoid the need for constructing long resolution chains. Now [(sort-of elephant)(Clyde), ¬animal(Clyde)] are compatible, but [elephant(Clyde), ¬(sort-of animal)(Clyde)] plainly are not. In fact the stronger statement can be made that [(sort-of elephant)(Clyde), ¬(sort-of animal)(Clyde)] are incompatible. (The statement is stronger because by the axiom schema for sort-of, it entails the incompatibility of [elephant(Clyde), ¬(sort-of animal)(Clyde)].) This incompatibility can again be efficiently detected with essentially the same lattice algorithms as were needed for the unhedged case, along with the general rule that if P is superordinate to Q in a type lattice, then (sort-of Q) is incompatible with ¬(sort-of P) (i.e., entails (sort-of P)).

For example (3), we observe that [(sort-of elephant)(Clyde), canary(Clyde)] and [elephant(Clyde), (sort-of canary)(Clyde)] are incompatible pairs. Given efficient methods for detecting incompatibility of type predicates P, Q, we can easily detect these new incompatibilities as well, using the rule that (sort-of P), Q are incompatible whenever P, Q are. Note, however that the stronger incompatibility observed in example (2) now fails: [(sort-of P)(x), (sort-of Q)(x)] are compatible even when P, Q are not; a thing can *conceivably* be both a sort of an elephant and a sort of canary (consider myths and fairy tales), though the actual existence of such a thing (even allowing prodigious advances in genetic engineering) may be wildly implausible.

The examples so far can inspire the hope that the data structures and algorithms we have already developed for efficient detection of superordination and incompatibility relationships in predicate taxonomies are sufficient as well for detecting incompatibilities of hedged predicates. This hope begins to falter, however, as we proceed to example (4). Curiously colour predicates, which we might imagine to be particularly "primitive" and more simply structured than nominal predicates, appear to obey more complex laws. Not only is it correct to say that [(sort-of tan)(Clyde), ¬(sort-of brown)(Clyde)] for example, are incompatible, as in the strong analog of example (2), but the strong analog of 4 now holds as well in certain cases; for example, if Clyde is sort of yellow (or yellowish) he cannot be sort of blue (or blueish). If this incompatibility held in all cases, there would still be no need for specialized representations of relationships among colour terms apart from those which can be captured in a simple specialization lattice. However, this is not the case; while a colour cannot be both sort of yellow and sort of blue, it *can* be both sort of yellow and sort of green, for example; in some sense, this is because yellow and green are more nearly compatible than yellow and blue.

One possible solution is to augment the basic hierarchy with special "sort-of" links. Ordinary links in a specialization (IS-A) hierarchy indicate subordination, and the direct descendants of a node are implicitly taken to be incompatible. The "sort-of" links would exhaustively specify for each colour whether it can be "sort-of" another colour.

One disadvantage would be the loss of the tree structure; for example, there would be two "sort-of" links connecting "chartreuse" to yellow and green respectively, which are on separate branches of the basic hierarchy. Moreover, there would be very many such links. The most serious problem, however, is that compatibilities and incompatibilities of *new* colour terms could not be predicted. For example, the mere absence of any colour term with "sort-of" links to both "blue" and "yellow" does not rule out the possibility that there *could* be a colour term with both links.

This has led us to consider an approach which introduces "strong incompatibility" links instead of sort-of links (fig. 1). Two colours P, Q are taken to be strongly incompatible just in case another colour cannot be both "sort-of P" and "sort-of Q". The resultant graph is rather pleasing and solves the problem of predicting incompatibility of hedged colour terms.

However, each new type of link introduced into a graphical representation of colours seems to capture only one type of relationship among colour terms. For example, the strong incompatibilities appear to provide no explanation of the intuition that if a colour is "sort-of scarlet" then it is not just "sort-of red", but simply red, whereas the analogous inference fails for "sort-of magenta". (Though magenta is a shade of red,

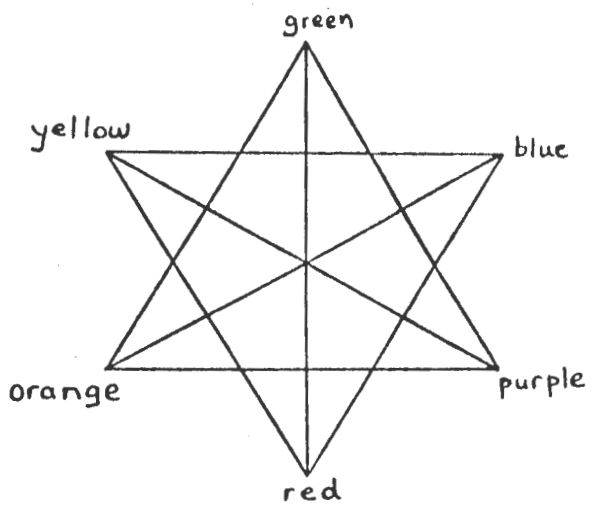


Figure 1: Strong incompatibility links between major colour terms. (The remaining 5 terms black, brown, grey, white and pink can be added as well).

a "sort-of magenta" may be too far towards the purple to be properly called red)

III. The cube model

Such subtleties have led us to a third kind of representation, more specialized still than hierarchies, which takes *quantitative* account of the (perceived) composition of colours. Those with experience in painting or computer graphics may be familiar with three dimensional models of colour space. One of the simplest models for representing colours as mixtures of primitives is the colour cube (fig. 2). Starting in

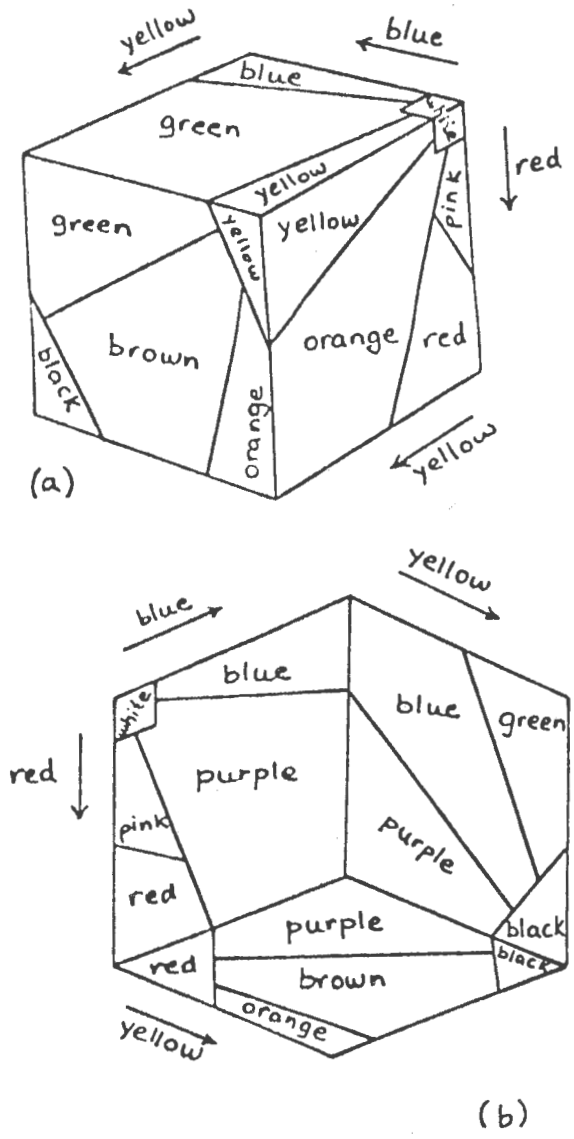


Figure 2 (a),(b): Two views of the colour cube

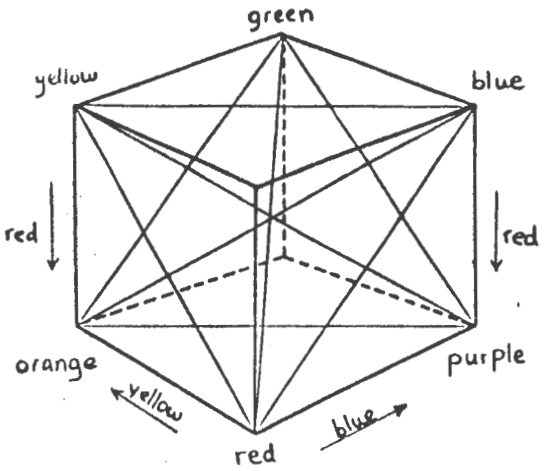


Figure 3: Strong incompatibility links seen as diagonals of the colour cube.

one corner (white) each of the edges coming out of that corner is increasing in the intensity of a primary. Any plane passing through the colour cube, parallel to one of the faces will be constant in one of the primaries and will contain all proportions of the remaining two. Thus the corners of the cube are: red, orange, black, purple, blue, green, yellow and white.

It is interesting to note that the star shaped graph of fig. 1, formed by the strong incompatibility relations among the major colours, can be embedded in the cube (fig. 3). Also, although there is a smooth transition between any two colours, a given colour term corresponds to a certain sub-volume inside the cube. It becomes apparent that a criterion for strong incompatibility is that the defining volumes for two predicates are not adjacent. Similarly, two colours are incompatible if the defining volumes are non-overlapping.

As one can appreciate from inspection of fig. 2, defining the subregions of the colour cube corresponding to the natural colours is not a trivial task. (We are not concerned about exact parameters, but we do need to do justice to the shapes of the regions and their interrelationships in order to achieve our inference objectives.) Instead of working directly with the primary values, we found it helpful to re-parameterize colours in terms of

$$\text{purity} = \frac{\text{pure colour component}}{\text{pure colour component} + \text{black component}}$$

$$= \frac{\text{pure colour component}}{1 - \text{white component}}$$

and

$$\text{dilution} = \text{white component}$$

where

$$\text{pure colour} = 1 - \text{white} - \text{black},$$

$$= \max(\text{red},\text{blue},\text{yellow}) - \min(\text{red},\text{blue},\text{yellow}),$$

white = 1 − max(red,blue,yellow),
black = min(red,blue,yellow),

and all quantities are assumed to range from 0 to 1.

A third quantity is needed to define the hue of the pure colour component. We will not go into details, since we have found the cube model to be non-optimal for our purposes. Despite its initial appeal, and although it is as adequate as any model for describing colours as additive or subtractive mixtures of three primaries, it fails to include all distinguishable colours. Surprising as this may seem, it is due to the fact that any three primaries can, at best, produce only part of the whole spectrum of hues (Judd 1963). Of course, any graphics tool that we can use to experiment with our model will employ three primaries, but nonetheless we prefer to choose a model that *in theory* could depict all distinguishable hues. Moreover, we would like regions defining the natural colour terms to be simpler than those in the cube model.

IV. The cylinder model

The cylinder model is similar in many respects to other well known models (Munsell 1969, Ostwald 1969) in that the hues are arranged in a circle around some axis and hue is specified by angular displacement. The essential difference from the cube model is that the chromatic (rainbow) hues are no longer viewed as composed of primaries, but simply as particular angular positions relative to a reference direction (say, scarlet). Purity and dilution, as defined earlier, are the other two dimensions. However, the "pure colour component" is no longer reducible to the difference between the max and min of the primaries, but simply represents the amount of single-hue colour which has been "blended" with certain amounts of black and white to produce the colour in question. Purity is 0 on the axis and increases radially to 1, and dilution increases vertically (i.e., axially) from 0 to 1. For purposes of graphical illustration, it is natural to shrink the top (white) surface of the cylinder to a point. In the resultant cone there are no paths of zero colour gradient (see fig. 4).

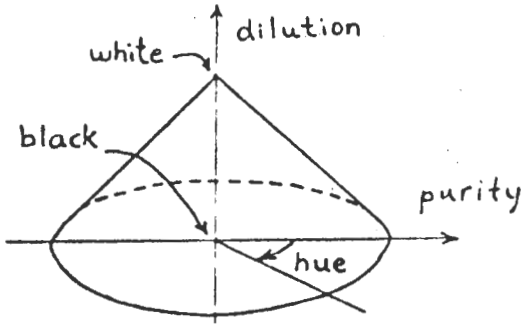


Figure 4: The purity-dilution-hue colour space, illustrated as a cone.

The cylinder model not only encompasses all colours, but in addition makes it possible to define colours as regions bounded by surfaces which are defined simply by keeping one coordinate fixed; i.e., colour regions are pie shaped portions of cylindrical annuli. Of course, in a representation which shrinks the top surface to a point, such regions taper towards the top (fig. 5).

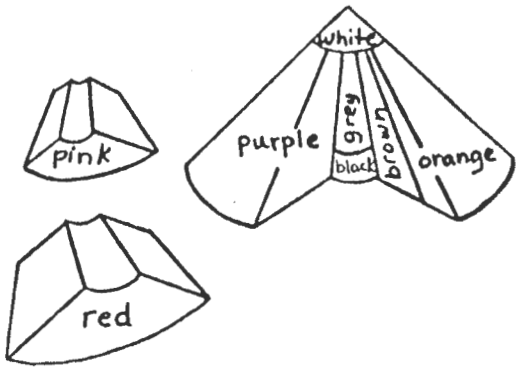


Figure 5: The cone model as partitioned by the basic colour regions. Here the red and pink regions are taken out to give a view of the interior of the solid.

V. Inference methods

As emphasized earlier, inference about colours is only a part of a general inference system. The goal is to design a special purpose mechanism that, given two literals — consisting of possibly hedged and/or negated colour predicates applied to unifiable arguments — will determine whether they can be resolved, i.e., it will indicate when the literals cannot both be true at once.

Going back to question 4 of the introduction, after resolving elephant(Clyde) against ¬elephant(x) of the second clause in the usual way, we are left with grey(Clyde) and yellow(Clyde) — these are the two literals supplied to the colour reasoning algorithm. In the model the predicates grey and yellow will correspond to non-overlapping regions so that grey(Clyde) can be resolved against yellow(Clyde) to yield the null clause, thus yielding "¬elephant(Clyde)", in a standard proof by contradiction.

The following examples illustrate the same type of reasoning as applied to a stucturally different kind of question:

1. *Given knowledge:* tan(x) V ¬elephant(x),
   ¬(sort-of brown)(Clyde),
   plus knowledge about colours
   Question: ?elephant(Clyde)

This states that all elephants are tan and that Clyde is not sort of brown. Since tan is a kind of brown it ought to be possible to prove that Clyde is not an elephant. Resolving the negation of this conclusion against ¬elephant(x) of the first clause we obtain tan(Clyde). The literals to be resolved by the colour reasoning algorithm are now "tan(Clyde)" and "¬(sort-of brown)(Clyde)". The defining region for tan is included in that for brown, so ¬(sort-of brown)(Clyde) can be resolved against tan(Clyde), thus disproving "elephant(Clyde)".

2 *Given knowledge:* grey(x) V ¬elephant(x),
¬(sort-of brown)(Clyde),
plus knowledge about colours
Question: ?elephant(Clyde)

This is very similar to question 1, but here there is not enough information to answer the question, since the resulting literals are grey(Clyde) and ¬(sort-of brown)(Clyde). It is clearly possible for Clyde to be grey and not sort of brown. Thus we cannot resolve these clauses.

The above examples indicate the need for:
1) a procedure *relation* which takes two colour predicates A and B and determines the relation of B to A, by comparing their hue, purity and dilution intervals,
2) a table which states whether for a given relation of B to A, and corresponding modes a, b (i.e., hedged and/or negated or simple, as given by the modifiers for A and B in the literals) the literals can be resolved.

mode a	mode b	relation between A and B: A B	A ⊃ B	A ⊂ B	A overlaps B	A = B
simple	simple	resolve	resolve			
simple	hedged	resolve				
simple	negated -hedged					
simple	negated					
hedged	simple	resolve				
hedged	hedged	resolve				
hedged	negated -hedged					
hedged	negated					
negated -hedged	simple		resolve	resolve	resolve	resolve
negated -hedged	hedged			resolve	resolve	resolve
negated -hedged	negated -hedged					
negated -hedged	negated					
negated	simple				resolve	resolve
negated	hedged					resolve
negated	negated -hedged					
negated	negated					

Figure 6: This table determines whether two literals can be resolved given the modes of each literal and the relation between the corresponding colour predicates. The relations, depicted graphically in the table are "apart", "adjacent", "overlapping", "included" and "centre-included".

The relation of B to A can be one of the following:
(i) "apart" – iff one or more of the corresponding intervals for A and B is not overlapping or adjacent
(ii) "adjacent" – iff all of the intervals are adjacent and possibly some but not all are included or overlapping
(iii) "overlapping" – iff all are overlapping and possibly some, but not all, are included
(iv) "included" – iff all of the intervals of B are included in A
(v) "including" – iff all of the intervals of A are included in B
(vi) "centre-included" – iff they are included and none of the corresponding intervals have common endpoints
(vii) "centre-including" – iff they are including and none of the corresponding intervals have common endpoints

The table will contain one entry each for "included" and "including" and for "centre-included" and "centre-including" and the order in which the inputs appear will determine the direction of the inclusion. Thus, when the table is queried the included predicate will always be in the position of B. This is to avoid repetitions.

The reason why we consider "included" and "centre-included" as separate cases is that they do result in different resolving patterns. For example, although both magenta and fire-engine red are special cases of red and their regions are thus included in that for red, anything that is "sort of fire-engine red" is clearly red, but something that is "sort of magenta" maybe too far towards the purple to be considered red.

Another distinction that must be made is between basic and non-basic colours. Detailed criteria for "basicness" are listed and discussed in Kay & McDaniel 1978, Mervis & Roth 1981 and Kay 1981. We take as basic the following 11 terms and no others: black, grey, white, red, orange, yellow, green, blue, purple, pink, and brown. We feel there is sufficient evidence in everyday usage to assume that these completely partition the colour space. Non-basic colours, such as yellow-green, navy, maroon, etc., sometimes lie across boundaries and in any case overlap one or more basic colours. Regions occupied by non-basic colours are generally smaller. A result of this is that *all* shades included by a non-basic term which spans two basic colours lie near the boundary between these basic colours. Thus, for example, all that is yellow-green is sort of yellow, yet not all that is yellow is sort of yellow-green.

These facts can be dealt with by pre-ordering literal pairs aA, bB before table look-up, in a manner dependent on the basic or non-basic status of A and B. Since basic colours always include non-basic ones, rather than vice-versa, and to be consistent with the order already established for the including case, the non-basic term, if any, is taken to be B. If both are basic or both are non-basic and the order is not forced by an inclusion relation, it is determined by ranking the literals in a prespecified manner dependent on their modes a, b, and taking the "lesser" of the two literals as aA and the "greater" as bB. The ordering we use for the input literal modes to the table is:

simple < hedged < negated-hedged < negated

The algorithm can be summarized as follows:

Given clauses aA(x) and bB(x) where A and B are colour predicates and a and b are either "sort-of", "¬", "¬sort-of" or nothing, which correspond to "hedged", "negated", "negated-hedged" and "simple",

respectively, first determine the relation between A and B.

Let
basic(x)=true iff x is one of the basic terms;
less(x,y)=true iff x precedes y, as described;

Then proceed as follows:

```
begin
r := relation(A,B);
ba := basic(A);
bb := basic(B);
if (r="centre-included" or r="included") then table(b,a,r);
if (r="adjacent" or r="overlapping") then
        if ( ba and bb) then table(b,a,r)
        else if (ba=bb and less((b,a)) then table(b,a,r)
else table(a,b,r);
end
```

VI. Conclusion

We conclude that there are classes of predicates which at first seem to require no more than quasi-hierarchical representations, but on closer examination are seen to call for more specialized representations of a quite different sort - in the case of colours, a numerically coded "spatial" representation. The cylinder model we have proposed allows constant time compatibility checking of various hedged and unhedged colour terms, minimizing the need for combinatorial inference. The model is also attractive for another reason: the colour cylinder could serve as the interface between the perceptual and conceptual systems of a robot; the required parameters should be quite easy to extract from the primary sensory data, and once extracted, could easily be used to compute an appropriate colour label.

Acknowledgments

We wish to thank Leo Hartman for his useful insights and disscussions of this paper. This work was supported by NSERC operating grant A8818.

References

[1] Birren, F. (editor) (1969), "Ostwald: The Color Primer". Van Nostrand Reinhold Co., New York.

[2] Birren, F. (editor) (1969), "Munsell: A Grammar of Color". Van Nostrand Reinhold Co., New York.

[3] Covington, A. R. and Schubert, L. K., (1979), "Organization of modally embedded propositions and of dependent concepts". Proc. of the 3rd Bienn. Conf. of the CSCSI/SCEIO, Victoria, B.C., May 14-16, 1980, pp 87-94.

[4] Grice, H. P., (1975), "Logic and conversation". In Davidson, D. and Harman, G. (editors), The Logic of Grammar, Dickenson, Encino, CA, pp 64-75.

[5] Judd, D. and Wyszecki, G., (1963). "Color in Business, Science and Industry" (2nd ed.). John Wiley and sons, New York.

[6] Kay, P. and McDaniel C. K., (1978), "The linguistic significance of the meanings of basic colour terms". Language 54, pp 610-146.

[7] Kay, P., (1981), "Colour perception and the meanings of colour words". Proc. of the 3rd Annual Conf. of the Cognitive Science Society, Berkeley, Calif., Aug. 19-21, 1981, pp 61-64.

[8] Mervis, C. B. and Roth, E. M., (1981), "The internal structure of basic and non-basic color categories". Language 57, pp 383-405.

[9] McSkimin, J. R. and Minker, J., (1979) "A predicate calculus based semantic network for deductive searching", Findler, N. V. (editor). Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, pp 121-175.

[10] Papalaskaris, M. A. and Schubert, L. K. (1981), "Parts inference: Closed and semi-closed partitioning graphs". Proc. 7th IJCAI, Vancouver, B.C., Aug. 24-28, pp 304-309.

[11] Schubert, L. K., (1979), "Problems with parts". Proc. 6th IJCAI, Tokyo, Aug. 20-23, pp 778-784.

[12] Schubert, L. K., (1976), "Extending the expressive power of semantic networks". Artificial Intelligence 7(2), pp 163-198.

[13] Schubert, L. K., Goebel, R. and Cercone, N., (1979). "The structure and organization of a semantic network for comprehension and inference". Findler, N. V. (editor), Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, pp 121-175.

The Representation of Presuppositions Using Defaults

R.E. Mercer

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5

R. Reiter

Computer Science Department
Rutgers University
New Brunswick, NJ 08903

ABSTRACT

This paper is a first step towards the computation of an inference based on language use, termed presupposition. Natural languages, unlike formal languages, can be semantically ambiguous. These ambiguities are resolved according to pragmatic rules. We take the position that presuppositions are inferences generated from these pragmatic rules. Presuppositions are then used to generate the preferred interpretation of the ambiguous natural language sentence. A preferred interpretation can be circumvented by an explicit inconsistency. This paper discusses the appropriateness of using default rules (Reiter(1980)) to represent certain common examples of presupposition in natural language. We believe that default rules are not only appropriate for representing presuppositions, but also provide a formal explanation for a precursory consistency-based presuppositional theory (Gazdar(1979)).

INTRODUCTION

The meaning of a natural language sentence includes the inferences that can be generated from the sentence together with knowledge about the world and knowledge about language use. One type of inference which can be generated in this manner is called a presupposition. What typifies this kind of inference is that both the sentence and its negation imply the same presuppositions. Originally proposed to infer the existence of a referent, it is now used to define those inferences, generated from a number of linguistic situations, which pass this negation test. The following sentences show some prototypical examples of presuppositions. In each of these examples the positive a-sentence entails and the negative b-sentence presupposes the c-sentence.

(1a) The present king of France is bald.
(1b) The present king of France is not bald.
(1c) There exists a present king of France.
(2a) Jack's children are bald.
(2b) Jack's children are not bald.
(2c) Jack has children.
(3a) Mary is surprised that Fred left.
(3b) Mary is not surprised that Fred left.
(3c) Fred left.
(4a) John stopped beating the rug.
(4b) John did not stop beating the rug.
(4c) John has been beating the rug.

This negation test led to one of the early definitions of presupposition:

> If A and B are sentences then
> A presupposes B iff
> (i) A entails B, and
> (ii) ¬A entails B.

It can easily be seen that under a bivalent semantics this definition leads to the unacceptable conclusion that B is a tautology. This observation subsequently led to attempts using multivalued semantics. Both Kempson(1975) and Gazdar(1979) give examples in which this semantics also fails to generate the appropriate presuppositions. In parallel with these attempts at a semantic definition of presupposition, there were candidates for pragmatic definitions as well. There were two sorts of presupposition suggested. First, speaker presuppositions are those that the speaker assumes the listener knows. Second, "plugs, holes, and filters" (categories of lexical items and the connectives and, or, and if...then, which stopped, permitted, and filtered out presuppositions) were considered in Karttunen(1973,1974) as explanations for the presuppositional behaviour of compound sentences. Both of these approaches are convincingly argued against in Gazdar(1979).

The persistent theme in these early attempts by linguists at defining presuppositions, as summarized in Kempson(1975), Wilson(1975), and Gazdar(1979) was that presuppositions are entailments of the sentence (and context in the case of the pragmatic definitions). Gazdar(1979) makes a major shift when he argues that presuppositions should be defined in terms of consistency rather than entailment. His arguments centre around the other main issue for linguists, the projection problem: given the presuppositions of a simple sentence, which ones survive the embedding of this sentence in a more complex sentence. More generally, how does the context affect a sentence's presuppositions.

An example of this contextual sensitivity follows: Under a "normal" interpretation, (4c)

in the example above can be "inferred" from (4b). But this inference is not an entailment since (4b) can be placed in a context which does not allow this inference.

(4d) John did not stop beating the rug because he hadn't started.

Hence, a presupposition of a sentence is consistent with that sentence, but when the sentence is placed in a larger context, the presupposition may be inconsistent; hence it can no longer be inferred. Presuppositions involve notions of incomplete knowledge and consequently non-monotonic systems of logic.

We essentially agree with Gazdar's approach; however we feel that his solution is somewhat ad hoc in that it is not a suitably formalized theory. In addition his theory uses a modal sentential logic. We prefer a first order representation of sentences. Also some of the questions that he poses cannot be answered within his framework, in particular: why are the lexical and syntactic sources of presuppositions as they are. In all fairness it should be pointed out that Gazdar(1979) is primarily interested in convincing the linguistic community of a certain pragmatic solution to the projection problem.

The main issue for us is the representation of presuppositions and the inferencing mechanisms required for generating these inferences in a coherent fashion. In doing so we feel that our representation can provide the extra insight needed to answer the above question. In particular, we view presuppositions in a more general sense: as inferences generated in the absence of complete knowledge. Our proposal is intended to provide the required formalism but keeps the essence of Gazdar's theory intact.

This paper presents a framework for representing presuppositions of asserted declarative sentences.

REPRESENTING PRESUPPOSITIONS USING DEFAULT RULES

This paper has a twofold purpose:

(1) to provide a computational mechanism for computing presuppositions,
(2) to furnish a formal explanation for a portion of the presuppositional theory in Gazdar(1979).

In reference to (1) the only other attempts to compute presuppositions of natural language utterances have been Joshi and Weischedel(1977) and Kaplan(1979). Since the algorithms contained therein are based upon a theory of presupposition (Karttunen(1973,1974)), which has been refuted by Gazdar(1979), these approaches are no longer serious candidates for computing presuppositions.

In reference to (2) the theory of Gazdar(1979) uncouples the generation of (potential) presuppositions from the checking of their consistency. In contrast, our theory represents each potential presupposition by a default rule. The proof theory for default logic provides the consistency checks required by Gazdar.

This is a novel use of default rules as a representational device. Reiter(1980) was motivated by a desire to represent beliefs about incompletely specified worlds. He also pointed out that default rules could be used to represent prototypical situations. The novelty of the current application is that we are using default rules to represent preferred interpretations of ambiguous linguistic forms.

A default rule is a rule of inference denoted

$$\frac{\alpha(\bar{x}) \ : \ M\beta(\bar{x})}{w(\bar{x})}$$

where $\alpha(\bar{x})$, $\beta(\bar{x})$, $w(\bar{x})$ are all first order formulae whose free variables are among those of $\bar{x} = x1,\ldots,xm$. Intuitively, a default rule can be interpreted as "For all individuals $x1,\ldots,xm$, if $\alpha(\bar{x})$ is believed and if $\beta(\bar{x})$ is consistent with our beliefs, then $w(\bar{x})$ may be believed". (Reiter(1980))

Some examples should point out the salient features. The first example will be given in some detail in order to describe the inferencing that leads to the preferred interpretation (Wilson(1975)) of an ambiguous lexical item (or syntactic construct in later examples). (Kempson(1975) uses the term natural interpretation.) Our solution is to represent the preferred interpretation of an ambiguous linguistic form as the inferences obtained as a result of deducing the consequent, $w(\bar{x})$, of a default rule. The formal definition of "a presupposition of a preferred interpretation" is then the consequent of a default rule.

Example 1 - Stop

In this example e represents an event, and t1 and t2 are time parameters meant to represent times relevant to the event, e. Even though a proper represention for continuous actions has yet to be obtained let us assume here that the following meets our requirements for a definition of "stop":

$$STOP(e) \longleftrightarrow (Et1\ t2).t1,t2 \ \& \ DO(e,t1) \ \& \ \neg DO(e,t2).$$

That is, for our purposes, an event stops iff there is a time t1 at which the event was being done and a later time t2 at which the event was not being done. We can then generate the definition of "not stop" by a simple negation to obtain

$$\neg STOP(e) \longleftrightarrow (t1\ t2).(t1,t2\ \&\ DO(e,t1)) \rightarrow DO(e,t2). \quad (*)$$

For this particular example the default rule for "not stop" would be

$$\frac{\neg STOP(e) : M(Et)DO(e,t)}{(Et)DO(e,t)}$$

This default rule now plays a crucial role in generating the preferred interpretation. If E is an event and ¬STOP(E) is given, for example

John did not stop beating the rug.

then using the default rule we can deduce

$$(Et)DO(E,t).$$

Using this inference, (*), and the given fact ¬STOP(E), we can also deduce

$$(Et).DO(E,t)\ \&\ (t').t,t' \rightarrow DO(E,t'),$$

that is, there is some time at which the event E was being done and it continues to be done at all future times. This matches our intuitions about the preferred interpretation of "not stopping E".

On the other hand,

John did not stop beating the rug because he was never doing it.

uses the "because clause" to indicate the extra qualification

$$(t).\neg DO(BEAT\text{-}RUG(John),t)$$

Using this qualification and (*) we can deduce

$$\neg STOP(BEAT\text{-}RUG(John)),$$

as required. Now the default rule cannot be invoked because its consistency condition is violated by the qualification.

Example 2 - Criterial and Noncriterial properties

In this example we look at a type of lexical presupposition which is based on the deciding criterion for a lexeme's meaning. Say then for purposes of this example, that the definition of "bachelor" is represented by the following first order sentence:

$$BACHELOR(x) \longleftrightarrow MALE(x)\ \&\ ADULT(x)\ \&\ \neg MARRIED(x)$$

Then the negation of "bachelor" would be:

$$\neg BACHELOR(x) \longleftrightarrow \neg MALE(x)\ V\ \neg ADULT(x)\ V\ MARRIED(x).$$

Thus if the knowledge base contained:

MALE(John)
ADULT(John)

and it was subsequently provided with the knowledge that

¬BACHELOR(John)

then it would be possible to derive MARRIED(John) from the definition of ¬BACHELOR(x). But if either of the first two formulae are absent the definition of ¬BACHELOR(x) is inadequate to deduce any of the remaining disjuncts. When used in a normal manner, however, "not a bachelor" typically means "a married adult male" whether or not ADULT(x) and MALE(x) are a priori knowledge. Being married or not married is then the typical criterion used to decide one's bachelor status. The noncriterial parts of the definition have been referred to as the presuppositions of the lexical item. Hence the noncriterial parts of the definition are entailed in the positive and presupposed in the negative uses of the lexeme. This knowledge of noncriterial parts of definitions of lexemes would thus be part of the knowledge base and could be represented as:

NONC(BACHELOR,MALE)
NONC(BACHELOR,ADULT)

In the same manner as Example 1, we capture the pragmatic rule for generating presuppositions in the following default rule schema:

$$\frac{\neg P(x)\ \&\ NONC(P,P1) : M\ P1(x)}{P1(x)}$$

Hence in situations where the age and sex of the non-bachelor are not known, the age and sex can be presupposed. For example

My cousin is not a bachelor.

would be represented as:

$$\neg BACHELOR(c1). \quad (*)$$

One instance of the default rule schema above is:

$$\frac{\neg BACHELOR(c1)\ \&\ NONC(BACHELOR,MALE) : M\ MALE(c1)}{MALE(c1)}$$

which would generate MALE(c1) and similarly a second instance would give ADULT(c1). Hence these are the presuppositions of ¬BACHELOR(c1). From (*), these two default consequences, and the definition of ¬BACHELOR, the desired MARRIED(c1) can be derived.

Example 3 - Factive verbs

Factives are a subcategory of verbs which can take a relative clause and which presuppose that clause, that is, normally imply the relative clause whether the verb is negated or not. For example

John regrets that Mary came to the party.
John does not regret that Mary came to the party.

Under normal circumstances, both of these sentences imply that

Mary came to the party.

We need the following axiom schema:

$$FACTIVE(P) \ \& \ P(x,\phi) \rightarrow \phi \quad (*)$$

where ϕ is any proposition. In addition, we propose the following default rule schema to provide the necessary presuppositions of factives:

$$\frac{FACTIVE(P) \ \& \ \neg P(x,\phi) \ : \ M\phi}{\phi}$$

where ϕ is a proposition. Suitable instances of these schemata for the above example would take REGRET for P, John for x, and COME(Mary,party1) for ϕ. Note that the knowledge base must contain the linguistic fact FACTIVE(REGRET).

Thus in its positive occurrence, the factive verb entails its complement, whereas in its negative, the factive verb presupposes its complement. This presupposition can be blocked in a context that blocks application of the default rule. For example

John does not regret that Mary came to the party because she didn't come.

Formally, we view this sentence as providing the information

$$\neg COME(Mary,party1). \quad (**)$$

From (**) and a suitable instance of the axiom schema (*) we are able to deduce

$$\neg REGRET(John,COME(Mary,party1)).$$

Moreover, the given fact (**) blocks the application of the default rule schema thereby preventing the derivation of the "normal" presupposition COME(Mary,party1).

Example 4 - Focus

Two methods of focusing parts of sentences which produce presuppositions are: (1) a syntactic method called clefting (clefts and pseudoclefts), and (2) an intonational method called contrastive stress.

Clefts and pseudoclefts. We do not define these two notions; instead we give an example which points out their important features.

Cleft of "John came (did not come).":
It was (not) John who came.
Presupposition: Someone came.

Pseudocleft of "John wanted (did not want) the dog.":
What John wanted was (not) the dog.
Presupposition: John wanted something.

Contrastive stress. Normal stress occurs at the end of a sentence, but any of the constituents in a sentence can be stressed with certain presuppositional consequences. If we have the normally stressed

Bill did not wreck this truck.

this could be represented as

$$\neg WRECK(Bill,truck1).$$

But we need some method for representing focused items wherever they occur. We will use λ-abstracted predicates as in Nash-Webber and Reiter(1977). The representations (disregarding tense) for the focused sentences are then:

Cleft: It was not John who came.
$\neg[\lambda x \ COME(x)]John$
Pseudocleft: What John wanted was not the dog.
$\neg[\lambda x \ WANT(John,x)]dog1$
Contrastive stress: Bill did not wreck this truck. (Underlining signifies stress.)
$\neg[\lambda x \ WRECK(x,truck1)]Bill$

We propose the following pragmatic rule:

$$\frac{\neg[\lambda x \phi(x)]u \ : \ M(Ey)\phi(y).}{(Ey)\phi(y)}$$

The presuppositions for each of the above focused sentences would then be generated appropriately. For example, given

Bill did not wreck this truck.

which is represented as

$$\neg[\lambda x \ WRECK(x,truck1)]Bill$$

we can derive the presupposition

$$(Ey)WRECK(y,truck1)$$

that is,

Someone wrecked this truck.

CONCLUSION

This paper regards presuppositions as inferences derived partly from pragmatic rules (conventions of language use), and discusses the suitability of using default rules to represent such rules. The preferred interpretation of an ambiguous lexical item or syntactic construct can then be inferred using the derived presuppositions.

ACKNOWLEDGEMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant nos. A3039 (to PC Gilmore) and A7642.

REFERENCES

Gazdar, G. (1979). Pragmatics: Implicature, Presupposition, and Logical Form, Academic Press.

Joshi, A.K. and R.M. Weischedel (1977). Computation of a Subclass of Inferences: Presupposition and Entailment, American Journal of Computational Linguistics, microfiche 63.

Kaplan, S.J. (1979). Cooperative Responses from a Portable Natural Language Data Base Query System, Stanford Heuristic Programming Project Technical Report HPP-79-19.

Karttunen, L. (1973). Presuppositions of Compound Sentences, Linguistic Inquiry 4, 169-193.

Karttunen, L. (1974). Presupposition and Linguistic Context, Theoretical Linguistics 1, 181-194.

Kempson, R.M. (1975). Presupposition and the Delimitation of Semantics, Cambridge University Press.

Nash-Webber, B. and R. Reiter (1977). Anaphora and Logical Form: On Formal Meaning Representations for Natural Language, Proceedings Fifth IJCAI, 121-131.

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

Wilson, D. (1975). Presuppositions and Non-Truth-Conditional Semantics, Academic Press.