

oml

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0310660 8

ORNL/TM-11938
(CESAR-91/32)

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

**Proceedings of the Sixth
International Symposium on
Methodologies for Intelligent
Systems (Poster Session)**

Karen S. Harber, Editor

September 1991

OAK RIDGE NATIONAL LABORATORY

CENTRAL RESEARCH LIBRARY

CIRCULATION SECTION

4500N ROOM 175

LIBRARY LOAN COPY

DO NOT TRANSFER TO ANOTHER PERSON

If you wish someone else to see this
report, send in name with report and
the library will arrange a loan.

UCRL-9119 (1-1991)

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11938
CESAR-91/32

Engineering Physics and Mathematics Division

**PROCEEDINGS OF THE SIXTH
INTERNATIONAL SYMPOSIUM ON
METHODOLOGIES FOR INTELLIGENT
SYSTEMS (POSTER SESSION)**

**October 16-19, 1991
Charlotte, North Carolina**

Karen S. Harber, Editor
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6364

DATE PUBLISHED — September 1991

Sponsors:
UNC-Charlotte
IBM-Charlotte
ORNL/CESAR

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



3 4456 0310660 8

Foreword

This volume contains papers which have been selected for the Poster Session at the Sixth International Symposium on Methodologies for Intelligent Systems - ISMIS'91, held in Charlotte, North Carolina, October 16-19, 1991. The Symposium was hosted by UNC-Charlotte and sponsored by IBM-Charlotte, ORNL/CESAR and UNC-Charlotte.

The Organizing Committee has selected the following major areas for ISMIS'91:

- * Expert Systems
- * Intelligent Databases
- * Knowledge Representation
- * Learning and Adaptive Systems
- * Logic for Artificial Intelligence.

These contributed papers have been selected from 55 full draft papers by the following Program Committee: A.W. Biermann (Duke), W. Bledsoe (Austin), J. Calmet (Germany), J. Carbonell (CMU), B. Chandrasekaran (Ohio State), P.R. Cohen (UM-Amherst), C. Fields (New Mexico State), B.R. Gaines (Canada), P.E. Hart (Syntelligence), S.J. Hong (IBM-Yorktown Heights), M. Karpinski (Germany), W. Kohn (Boeing, Seattle), K. Konolige (SRI), C. Lassez (IBM-Yorktown Heights), R. Lopez de Mantaras (Spain), J. Maitan (Lockheed), R.A. Meersman (The Netherlands), R. Michalski (George Mason), J. Minker (Maryland), M. Mukaidono (Japan), R. Parikh (CUNY), J. Pearl (UCLA), D. Perlis (Maryland), F.G. Pin (ORNL), H. Prade (France), Z.W. Ras (UNC-C), L. Saitta (Italy), E. Sandewall (Sweden), T. Sellis (Maryland), J. Sowa (IBM-Yorktown Heights), R. Thomason (Pittsburgh), D. Touretzky (CMU), R. Waldinger (SRI), S.K.M. Wong (Canada), M. Zemankova (NSF) and J. Zytow (Wichita State). The activity of this Committee and all of the cooperating referees was a great help in completing the final program. This help is highly appreciated.

The cooperating referees are listed below:

J. Baker, B. Chu, L. Console, M. Franco, H. Geffner, A. Giordana, L. Giordano, J. Grzymala-Busse, M. Maher, A. Martelli, S. Matwin, E. Mays, Z. Michalewicz, E. Plaza, H. Rasiowa, P. Torasso, J. Xiao, R. Yap and W. Zadrozny.

The Symposium has been organized by the University of North Carolina at Charlotte with the following Organizing Committee: Bill Chu (UNC-C), Karen S. Harber (ORNL), Zbigniew Michalewicz (UNC-C), M.S. Narasimha (IBM-Charlotte), Francois G. Pin (ORNL), Zbigniew W. Ras (Symposium Co-Chair, UNC-C), Jing Xiao (UNC-C), Maria Zemankova (Symposium Co-Chair, NSF).

We wish to express our thanks to Alan Biermann, Jon Doyle, Larry Kerschberg, Tom Mitchell, and Gio Wiederhold who gave invited talks at ISMIS'91. We would also like to express our appreciation to ISMIS'91 sponsors, to all who submitted papers for presentation at the symposium and publication in this proceedings, to ISMIS'91 Organizing Committee, to Karen Harber at ORNL without whose help the present volume could not have been completed and to all of those who contributed to the symposium program.

Francois G. Pin
Zbigniew W. Ras
Maria Zemankova

September, 1991
Charlotte, N.C.

TABLE OF CONTENTS

Robotic Mobility and Cognitive Maps Carl M. Benda	1
A Quantitative Analysis of Reasoning for RMS Laurent Buisson and Jérôme Euzenat	9
A Strategy for the Computation of Conditional Answers Robert Demolombe	21
Integrated Reasoning Through Associative Retrieval Thomas C. Eskridge	33
A First Order Theory for Representing Space in Knowledge-Based Scene Generation Enrico Giunchiglia and Alessandro Armando	45
Learning Intermediate Concepts in Causal Trees: A Direct Method Jean-Louis Golmard	57
Transformation-Ordering Iterative-Deepening-A* Lawrence O. Hall, Diane J. Cook, and Willard Thomas	65
An Improvement of Weighting Strategy in Resolution-Based Automated Reasoning Yong-Gi Kim and Ladislav J. Kohout	73
Pattern Recognition Using the Third and the Fifth Classes of Dynamical Systems Ying Liu and Hede Ma	83
Sensory Integration G. T. McKee and E. T. Powner	95
Data Structures + Genetic Operators = Evolution Programs Zbigniew Michalewicz, Joseph Schell, and David Seniv	107
Generic Problem Solving Models in a Computer Aided Methodology for the Construction of Knowledge Based Systems André R. Probst, Alex I. Horvitz, and Dieter Wenger	119
Generation of Production Rules by Backward Search from Neural Networks Da Qun Qian, Piero Scaruffi, and Dario Russi	131
Computational Nonmonotonism and the Qualification Problem Arcot Rajasekar	143

Task Allocation by a Team of Learning Automata	
Franciszek Seredynski	155
Notes on Rough Inference	
Brian Shay	167
Bottom-Up Evaluation in Indefinite Deductive Databases	
Rajshekhar Sunderraman	179
Inheritance in Conceptual Networks	
Leixuan Yang and Stan Szpakowicz	191
Combining Symbolic and Numeric Representations in Learning Flexible Concepts: the FCLS System	
Jianping Zhang	203

POSTER SESSIONS

APPROXIMATE REASONING

“An Improvement of Weighting Strategy in Resolution-Based Automated Reasoning,” Yong-Gi Kim and Ladislav J. Kohout (Florida State Univ.)

“Data Structures + Genetic Operators = Evolution Programs,” Zbigniew Michalewicz, Joseph Schell, and David Seniv (UNC-Charlotte)

“A Quantitative Analysis of Reasoning for RMS,” Laurent Buisson and Jérôme Euzenat (France)

EXPERT SYSTEMS

“Generic Problem Solving Models in a Computer Aided Methodology for the Construction of Knowledge Based Systems,” André R. Probst, Alex I. Horvitz, and Dieter Wenger (Switzerland)

INTELLIGENT DATABASES

“A Strategy for the Computation of Conditional Answers,” Robert Demolombe (Toulouse, France)

“Bottom-Up Evaluation in Indefinite Deductive Databases,” Rajshekhar Sunderraman (Wichita State Univ.)

KNOWLEDGE REPRESENTATION

“Integrated Reasoning Through Associative Retrieval,” Thomas C. Eskridge (New Mexico State Univ.)

“A First Order Theory for Representing Space in Knowledge-Based Scene Generation,” Enrico Giunchiglia and Alessandro Armando (Genoa, Italy)

“Inheritance in Conceptual Networks,” Leixuan Yang (Ottawa, Canada) and Stan Szpakowica (Johannesburg, South Africa)

“Sensory Integration,” G. T. McKee (Univ. Reading, England) and E. T. Powner (Univ. Manchester, England)

“Pattern Recognition Using the Third and the Fifth Classes of Dynamical Systems,” Ying Liu and Hede Ma (Savannah State College)

“Robotic Mobility and Cognitive Maps,” Carl M. Benda (IBM-Charlotte)

LEARNING AND ADAPTIVE SYSTEMS

“Task Allocation by a Team of Learning Automata,” Franciszek Seredynski (Warsaw, Poland)

“Generation of Production Rules by Backward Search from Neural Networks,” Da Qun Qian, Piero Scaruffi, and Dario Russi (Olivetti A.I. Center, Italy)

“Combining Symbolic and Numeric Representations in Learning Flexible Concepts: the FCLS System,” Jianping Zhang (Utah State Univ.)

“Learning Intermediate Concepts in Causal Trees: A Direct Method,” Jean-Louis Golmard (Paris, France)

LOGIC FOR AI

“Computational Nonmonotonicism and the Qualification Problem,” Arcot Rajasekar (Univ. of Kentucky)

“Notes on Rough Inference,” Brian Shay (Hunter College, CUNY)

METHODOLOGICAL ISSUES

“Transformation-Ordering Iterative-Deepening-A*,” Lawrence O. Hall, Diane J. Cook, and Willard Thomas (Univ. of South Florida)

Robotic Mobility and Cognitive Maps

Carl M. Benda
IBM
Charlotte
Charlotte, NC 28257

Abstract

This paper focuses on the acquisition and storage of environmental information by a mobile robot. The environment includes obstacles which must be mapped by the robot. A method for representing a robot moving through its environment is described. Algorithms used for storage and retrieval of environmental data are presented. The goal is to represent the data in what is known as a cognitive map which allows faster retrieval of the environmental data because of this representation's compact nature.

1. Introduction

The paper focuses on tools and algorithms used to show how a mobile robot could map out an area of his environment and store that map for later use. The environment of the robot contains various obstacles which will be stored in the cognitive map of the environment. Specifically, this paper will focus on three main areas of the project:

1. Circular Quadtree Data Structures ([3], [4])
2. Depicting the Data in Polar Coordinates ([3], [4])
3. Creating a Cognitive Map

The robot, by processing information obtained through the use of these mapping algorithms can detect exactly where obstructions are. This paper will present the software structure and algorithms employed to generate the final goal, which is a "Cognitive Map" [6] of the original environment information.

2. Circular Quadtree

The "Quadtree Data Structure" as described in [2,3,4,5] is used to store the information that has been scanned. There are in fact many methods available for storing environment information, from simple memory dumps to complex compression algorithms. The quadtree data structure provides a useful yet efficient method of storing environmental information. Figure 2.1 graphically depicts the method used by the robot to store the scanned data and the order in which the data is examined.

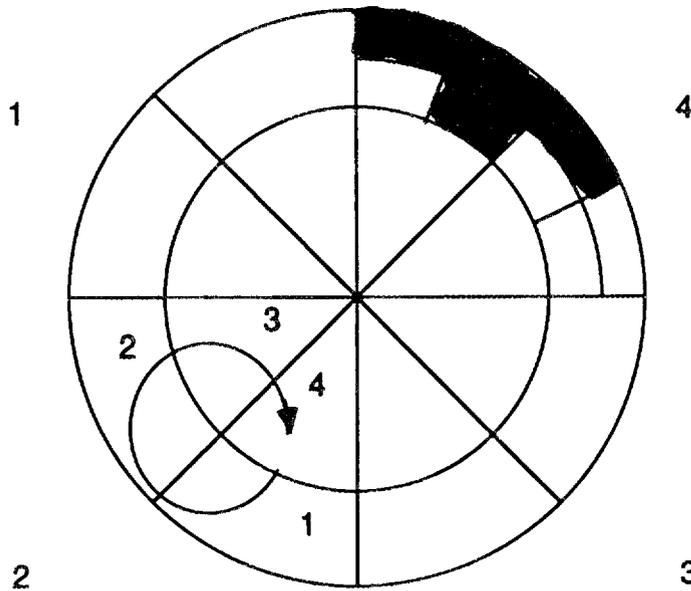


Figure 2.1. Circular Quadtree

In the quadtree data structure, elements are assigned values depending upon the data contained within these elements. The elements have a value of either 0, 1, or 2. A value of zero means that for the sector scanned, there is no environmental information. To put it another way, the environment contains no obstacles in that particular sector. A value of 1 means that for the particular sector, the environment contains an obstacle which must be avoided. For a value of 2, the environment sector being scanned must be subdivided into four subsectors to further determine the extent of the object. These subsectors are then recursively scanned in a counter clockwise fashion. This recursive algorithm continues until the subsector is found to contain an object or be devoid of an object. When all four quadrants have been scanned, the resulting quadtree represents all of the visible local environmental data. A tree is created to represent the storage of the environmental information. Figure 2.2 shows how the data structure in Figure 2.1 is stored in a hierarchical format.

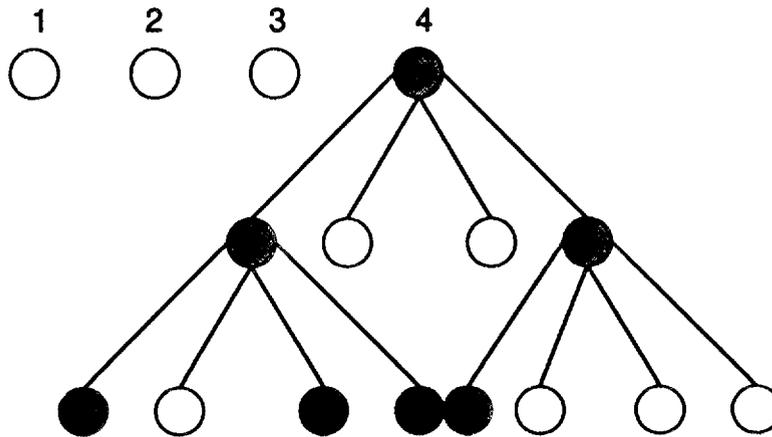


Figure 2.2. Circular Quadtree Hierarchical Structure.

3. Scanning Technique

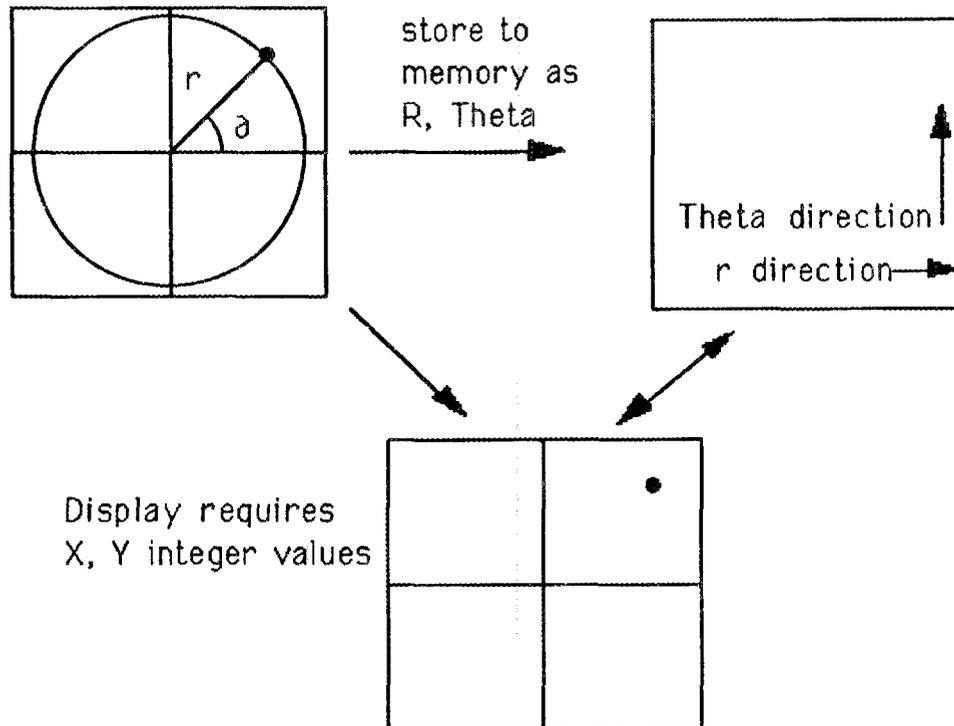
The first element to be determined is the scanning radius of the robot. After the radius has been set, the scanning method must be chosen. The direction of the scan is not as important as the final quadtree, as any direction would produce an equivalent quadtree.

3.1 Data Transformation

The scanned environment is stored as R, Theta polar coordinates. To display the location of the objects scanned, the data is translated into X,Y coordinates which the display hardware uses. See Figure 3.1.

3.1. Transformation from polar coordinates to positive integer Cartesian coordinates takes place using a simple software algorithm.

Scanned as R, Theta



Coordinate Translation Usage

Figure 3.1. Data Representaions

3.2. The Scanning and Subdivision Algorithms

In order to be as efficient as possible, the scanning algorithm only scans until the area of concern is known to contain an object. In this way, the subdivided area is completely scanned only when it is devoid of an object. Using a single subsector, Figure 3.2 depicts the scanning sequence used by the algorithm. Once the area of detail is *resolved*, the algorithm stores the location of the subsector in a table. This information is stored along with the level of resolution required to resolve the subsector. The level of resolution is the number of subdivisions required to ascertain whether the subsector in question is devoid of an object or filled with an object. In practice however, as will be shown, the number of subdivisions is limited to the ability of the machine to translate from polar coordinate information to integer relative points in the environment. After four subdivisions within a given sector, if the subsector is still known to be gray, the robot stores that information into the table. Because subdivision is done using polar coordinates, the areas close to the center of the environment can not be resolved beyond the fourth level. The level of subdivision is the number of times the subdivision algorithm calls itself within a given subsector of the environment.

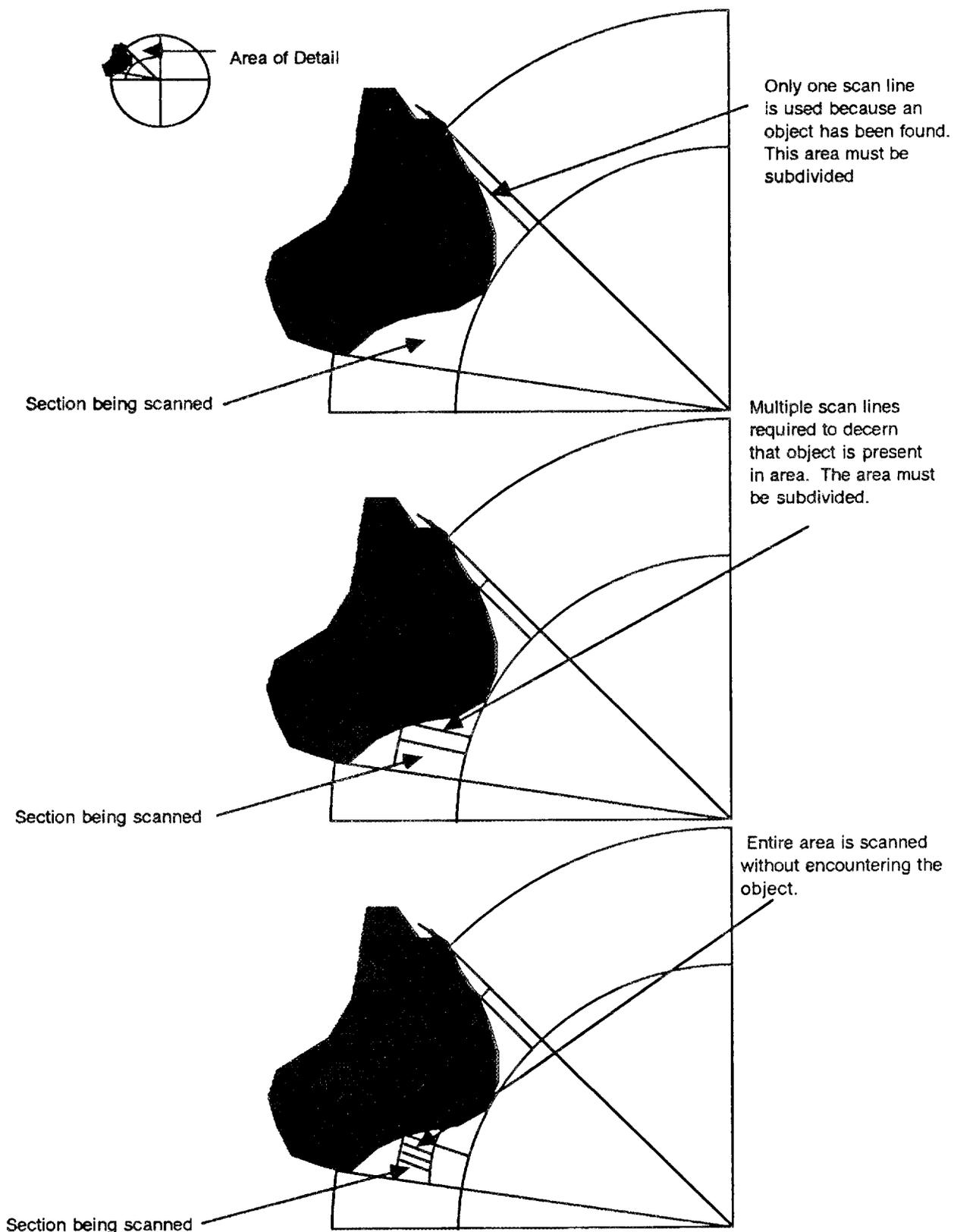


Figure 3.2. Scanning Example

Information is stored in memory in a simple format. The data consists of level, color, starting angle, ending angle, and two radial values, (one indicating the starting radius, and the second indicating the final radius of the subsector, both outward from the position of the robot). The level is a number from 1 to 4. The color is stored as an ASCII character which is one of either b, w, or g. The radius values are stored as a relative distance value from 0 to 240. For example, in figure 3.2, the sector being scanned in the final view would have a starting radius value of 210, an ending radius value of 240, and a starting angle of 168.

The major difficulty in scanning the environment is that all of the points on the environment are in reality only addresses in memory, and thus can be referenced only in an integer fashion. The scanning process, however, requires floating point calculations in order to more realistically depict the vision process. The floating point calculations are handled with the float data structure in C. In view 3 of Figure 3.2, using the above formula, the calculation for the starting angle works out to be 168.75, but because it is evaluated to an integer, the starting angle stored onto the would be 168 degrees. In application this approximation performs satisfactorily. Moreover, when an algorithm which incorporated rounding was used, it proved not to add a significant change in the overall accuracy of the robot. However, the more complicated algorithm did add quite a lot of overhead to the program thus slowing execution down significantly.

Determining how and when to subdivide the current sector if it does contain an obstacle is the job of the subdivision algorithm. Starting with the current position of the sector that is being scanned, the subdivision algorithm determines where the next subsector to be scanned is located. The key to the successful completion of the subdivision algorithm is its use of recursion. One point not shown, however, is the fact that there is a practical limitation to the amount of subdivision that can be done. In theory, subdivision of the area can continue to infinity. The practical limitation is based solely on the resolution of the environment. Through trial and error it was found subdivision would be allowed to continue as long as necessary or until a depth of 4 was reached. Figure 3.3 is a quadtree representation of a single quadrant showing that an end node does not have to be either all black or all white, but because a depth of four has been reached no further subdivision will take place on that subsector.

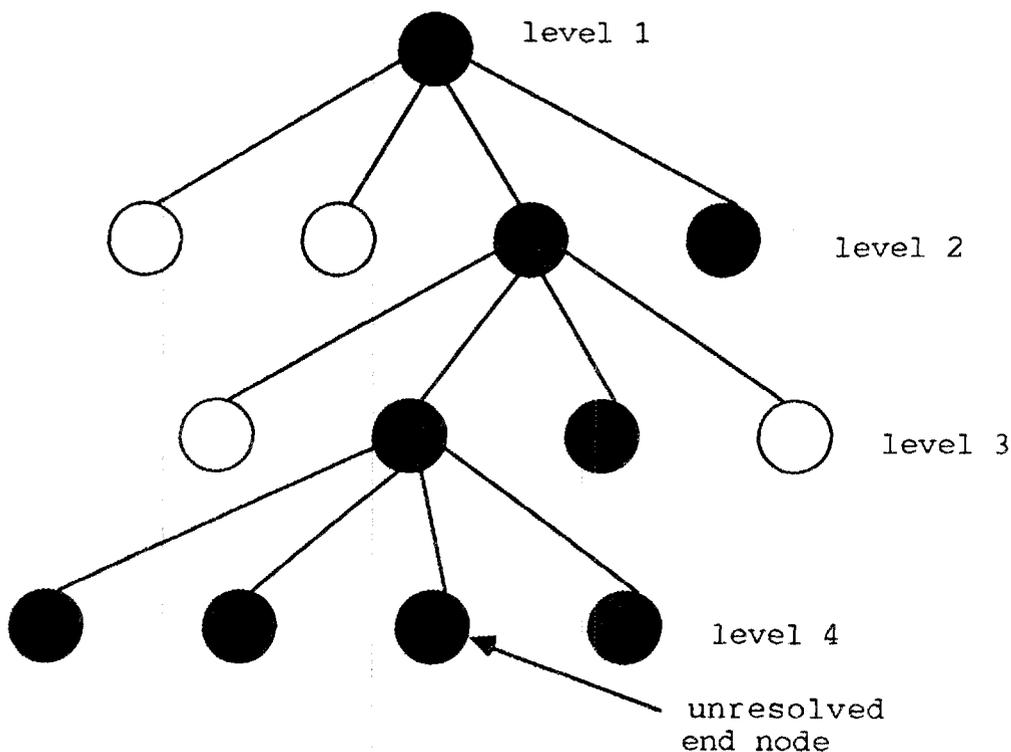


Figure 3.3. Quadtree with an Unresolved End Node.

4. Robotic Mobility and Creating the Cognitive Map

The study of cognitive maps is not new, but has been around since the 19th century [6]. In this paper, cognitive maps are used to create a data base for information used to show where the Robot is in relation to the objects that are to be avoided.

4.1. A Representation of the Cognitive Mapping Process

The cognitive mapping process establishes a criterion for connecting raw sensory data, in this case the scanned-in data represented in R/Theta format, into objects. A Cognitive map is a representation of the environment in connected objects. There has been much activity in the area of cognitive maps coming from a wide variety of disciplines such as urban planning and computer science. The theory is relatively well understood. For the purposes of clarification, a *Cognitive Map* is the spatial representation of objects within the scope of the viewer's domain [7]. In this work, the viewer is the *robot*, and its domain is a radius of 240 relative points, 360 degrees. This domain is known as the scanned environment. There are many aspects of an environment that may need to be made an explicit part of the Cognitive Map including color and spatial layout of the surfaces of any objects in the environment. The layout of the 2 dimensional object is used to create the Cognitive Map. In this section an algorithm is introduced that will collect the data used to create the Cognitive Map. With respect to the environment, there are three elements that must be decided upon. Fortunately, due to the nature of the design of the quadtree data structure, and the scanning and subdivision algorithms, two out of three of these elements have been decided already. However, for purposes of edification, the three elements are presented here.

1) Primitive elements [8] are used for representing the shape of objects within the domain of the system. The point here is that although the domain of the robot continues as stated above, with the ability the robot to travel, the domain of experience grows as the robot moves within the environment. Typically, the primitive elements may be one of either a limited local descriptor or a more general area descriptor, and since the concern is to compute the spatial layout, i.e. the Cognitive Map, of the environment, the choice is to use an area descriptor. An area descriptor describes the space in which the robot is traveling [6]. Attributes of this space include Empty Space and NON-Empty Space. In Figure 5.1 below, the robot's initial perception of its environment is shown. The lines represent the robots line of sight, and the polygonal shapes are objects which the robot cannot see past.

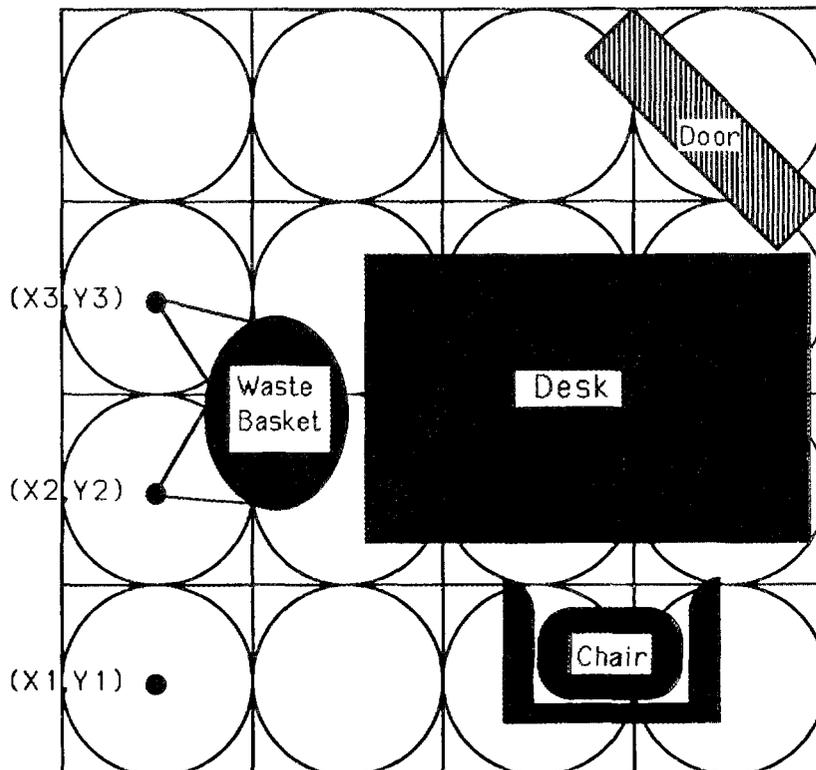


Figure 4.1. Initial Space Representation for the Robot.

2) The second element which must be decided upon is the choice of coordinate system. In [6], the discussion concludes with the author selecting a nonegocentric framework. This means that since the world is stable and it is the robot that keeps changing position on the environment, the Cognitive Map should be computed independent of the perceiver's point of view. Originally, the quadtree data structure representation of the objects on the environment had been pre-calculated.

3) These first two elements, although important in themselves, are more or less a means of coming to the third element of the Cognitive Map, the organization. There are two possible alternatives for organizing the primitive elements to represent the structure of the environment. The first is to describe the relationship between individual objects as they are perceived, and the second is to partition the objects into groups of objects which are then connected together as a whole.

4.2. Computing the Cognitive Map

To compute the entire data base, referred to as the domain of experience for the robot, an algorithm has been generated which collects environmental data based on the quadtree recreation of the environment. A set of cognitive maps are produced which are written to tables. These tables collectively are known as the domain of experience for the robot. The path, along with the environmental data is entered by the robot. The robot travels at a known rate, and along the way stops to obtain new cognitive mapping data.

Collected data is stored in a table. The data describes at what angle the object was first observed to be located and the relative distance from the robot. After scanning a complete 360 degrees, the robot then moves a specified distance along the selected path and repeats the scanning process at the new location.

4.3. How the Cognitive Map is Created.

The objective of depicting a Cognitive Map for environmental representation is to provide a comprehensive depiction of whether or not the planned path is viable. The idea of mapping out exactly where the objects in the environment are located is not as important for robot navigation as showing a blocked path.

Using the Cognitive Map, the robot is able to avoid areas where obstacles exist. The Cognitive Map defines regions of the environment which are not passable, rather than storing the size and location of the obstacles. The data used to create the Cognitive Map is stored in a Table .

A Cognitive Map is an important tool for the use of robot navigation when the goal is to achieve navigation to a destination point. Clearly if the goal is to gather data about the environment for later study, as is the case with a probe, the Cognitive Map as it is employed here would require modification. Perhaps the best reason for creating a Cognitive Map for data representation of environmental data can be summed up in one word; Speed. If the relative sizes measured in bytes of the R/Theta table and a randomly selected Cognitive Map were compared, the R/Theta data table would be an order of magnitude larger than that of the Cognitive Map. Just like the R/Theta data table, each data table contributing to the Cognitive Map is used to show where the objects in the environment are located. This means that the amount of data that the robot must consider, to make a decision about which direction to travel, is far less if the Cognitive Map is used rather than the R/Theta data table. It should be kept in mind that the development of the Cognitive Map requires that the R/Theta representation be read and manipulated at least one time. If the robot moves into a new region, which has not been previously scanned, additional scanning of this new region would be required. If, however, one considers the complexity of the data represented by the two formats, it is also clear that the Cognitive map has less complex information.

The advantage to the R/Theta data table is that it gives an additional view of the entire scanned-in data. It provides information on where the environment has subsectors that are gray, black, and white. In the Cognitive Map, the only information that matters is the visible contour the obstacles, i.e. how far away they are, and in what direction they lie.

The Cognitive Map is a simple way to get around the burden of knowing everything about all of the environment. It relies on the R/Theta mapping process, but really only once.

References

1. Snyder, W. E., *Industrial Robots: Computer Interfacing and Control* (Prentice Hall Inc. 1985).
2. Samet, H., *Data Structures for Quadtree Approximation and Compression* (Department of Computer Science University of Maryland, 1982).
3. Chen, S., *Spherical Data Structure and Visual Feedback for Robotic Control* (Proc. First Annual Workshop on Intelligent Control, Troy, NY August, 1985).
4. Chen, S., *Multi-sensor Fusion and Navigation of Mobile Robots* (Special Issue on Robotic Navigation, International Journal of Intelligent Systems, 1987).
5. Yau, M. and Srihari, S., *A Hierarchical Data Structure for Multidimensional Digital Images* (Communications of the ACM July, 1983).
6. Yeap, W. K., *Towards a Computational Theory of Cognitive Maps* (Artificial Intelligence Number 3, 1988).
7. Huttenlocher, J. and Presson, C. C., *The Coding and Transformation of Spatial Information* (Cognitive Psychology Number 11, 1979).
8. Marr, D. and Nishihara, H. K., *Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes* (Proc. Roy. Soc. B 200, 1978).
9. Nishihara, H. K., *Intensity, Visible-Surface, and Volumetric Representations* (Artificial Intelligence Number 17, 1981).

A QUANTITATIVE ANALYSIS OF REASONING FOR RMS

Laurent Buisson and Jérôme Euzenat

Division Nivologie	IRIMAG/INPG
CEMAGREF	Laboratoire ARTEMIS/Imag
BP 76	BP 53 X
F-38402 SAINT-MARTIN D'HERES	F-38041 GRENOBLE Cedex
internet: Jerome.Euzenat@sherpa.imag.fr	— uucp: euzenat@imag.fr

ABSTRACT

For reasoning systems, it is sometime useful to cache away the inferred values. Meanwhile, when the system works in a dynamic environment, cache coherence has to be performed, and this can be achieved with the help of a reasoning maintenance system (RMS). The questions to be answered, before implementing such a system for a particular application, are: how much is caching useful ? Does the system need a dynamicity management system ? Is a RMS suited (what will be its overhead) ?

We provide an application driven evaluation framework in order to answer these questions. The evaluation is based on the real work to be processed on the reasoning of the application. First, we express the action of caching and maintaining with two concepts: backward and forward cone effects. Then we quantify the inference time for those systems and find the quantification of the cone effects in the formulas.

1. INTRODUCTION

For reasoning systems such as knowledge bases, it is often necessary to record the result of the inference process even if it is goal driven. Recording the result of a computation is called *caching* in computer science. Caching is necessary when the produced inferences are costly and used several times.

When knowledge in the base does not evolve, caching is safe and very efficient. But in real world applications, the knowledge base is usually dynamic. This is true for systems that interact with the environment (through sensors) or with the user who can set hypotheses and change the knowledge in the base. So, caching requires dynamicity management. Most of the time, it is performed by using a RMS (Reasoning Maintenance System) based on dependency graph manipulation. But is a RMS always interesting ? Should it be more attractive to treat dynamicity problems by ignoring RMS solutions ?

We develop here a quantitative analysis of the reasoning graph in order to answer these questions. Numeric criteria defined on properties of the dependency graph are used. Real world applications give evidence of such properties, especially for spatial knowledge bases and spatial reasoning.

After a short description of reasoning maintenance systems and their advantages in the context of knowledge bases, we will briefly describe an object-based knowledge

base management system called Shirka/TMS which uses a RMS (§2). We will show some numeric results from that system and give expectations about its behavior. More recently, observations have been performed on a real world application ELSA (§3) dedicated to the analysis of snow avalanche path. This application uses the inference mechanisms in order to compute spatial properties of a geographic area such as *connected* sub-areas or *close* ridges which are used very often. The benchmark results obtained with ELSA are very surprising.

We are able to explain them with the help of a new concept: backward and forward cone effects. They are formalized (§4) in order to draw general conclusions about RMS use in reasoning systems. In fact, the advantage of a RMS toward rough caching is a tradeoff between backward and forward cone effect.

2. A SPATIAL REASONING APPLICATION

The motivations for using a RMS in knowledge based systems are first presented. Then, Shirka/TMS will be introduced together with some tests and expectations about its behavior.

2.1. REASONING MAINTENANCE SYSTEM

When using an inference system in backward chaining mode, the result of each inference, would it be an attribute value or the validity status of a proposition, can be cached i.e. recorded in memory. Cached values do not have to be inferred twice or more. In fact, caching is useful when a value is used several times by the system and is as useful as the number of times the value is needed. But, while caching uses additional memory space and time, it has to be used with care.

Moreover, in evolving systems or when the inferences allowed by the system can be nonmonotonic, something which is considered as holding (a value considered as the value of an attribute or a proposition considered as true) can be discarded. In such cases, the cached values must be invalidated, i.e. not cached anymore. This is the job of a RMS.

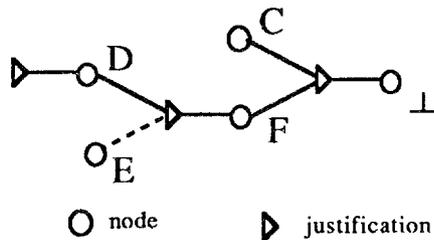


Fig. 1. A dependency graph is here represented with circles as nodes and triangles as justifications where the nodes in the IN-list come through a full line while nodes in the OUT-list come through a dotted line. Nodes that have a justification whose IN- and OUT-lists are empty (e.g. D) represent true formulas because they do not need to be inferred.

Reasoning maintenance systems (RMS) are aimed at managing a knowledge base considering different kinds of reasoning. Such a system is connected to a reasoner (or problem solver or inference engine) which communicates every inference made. The RMS has in charge the maintenance of the reasoner's current belief base. RMS developed so far focussed on nonmonotonic reasoning or multiple contexts reasoning. They record each inference in a *justification* that relates *nodes* representing propositional formulas plus a special atom (\perp) representing contradiction. A justification $\langle \{i_1, \dots, i_n\} \{o_1, \dots, o_m\} \rangle: c$ is made of an IN-list ($\{i_1, \dots, i_n\}$) and an OUT-

list $(\{o_1, \dots, o_m\})$. Such a justification is said to be valid if and only if all the nodes in the IN-list are known to hold while those in the OUT-list are not; a node, in turn, is known to hold if and only if it is the consequent (c) of a valid justification. The recursion of the definition is stopped by nodes without justification and by the axioms that are nodes with a justification containing empty IN- and OUT-lists.

2.2. SHIRKA/TMS AND ITS BEHAVIOR

Shirka is a traditional object based knowledge representation system written in Lisp [1]. Everything, in Shirka, is an object (including inference methods...). Each object belongs to a class which defines its structure — in terms of a list of fields and constraints on the fields values — and its inferential capabilities — in terms of inference methods used in order to determine the values of unfilled fields. Inference methods are among value passing, procedural attachment, pattern matching and default values.

Classes are organized in a direct acyclic graph structured by the *a-kind-of* relationship between classes. This relationship enables inheritance from a class to its specializations. Inheritance is used through class refinement — a class strongly inherits, i.e. possesses, its constraints on fields from its super-class — and inference specialization — a class weakly inherits, i.e. inherits by default, its inference methods.

A RMS has been implemented on Shirka. It is standard except that it records and propagates field values [2]. The underlying assumption of the implementation of a RMS in an object-based knowledge representation is that the base is queried very often (or not often modified). The performances are very attractive because re-infering is avoided (and so, the answers are given very quickly). On another hand, the modifications — that are safely dealt with — and initial inferences are processed more slowly. This assumption was enforced by the observations made with the very simple tests below.

2.3. ELSA: A SPATIAL REASONING APPLICATION

In the context of spatial reasoning, the RMS is very attractive. In other words, spatial reasoning appears as a good application domain. Meanwhile, some effects which have not been presented yet can be observed in that kind of applications: they are “forward and backward cone effects”. These observations were performed on a real world application dedicated to the analysis of snow avalanche paths: ELSA.

We first present ELSA and the advantage of using a RMS in the context of spatial reasoning. Then, a set of numeric tests are discussed which demonstrates the advantage of using a RMS in ELSA. At last, those results are summarized in two principles called backward and forward cone effect.

ELSA is a problem solving environment which offers to a snow specialist the different tools available in order to perform an avalanche path analysis and choose the best protection devices. As it has been explained elsewhere [3, 4], ELSA is built on Shirka/TMS. ELSA is a knowledge based system which uses both symbolic simulation based on expert knowledge and numerical simulation based on fluid mechanics conservative laws.

Because of the spatial extension of the phenomena involved in snow avalanches (snow-drift, snow-cover stability, fracture propagation, avalanche flowing...), ELSA needs spatial information on the path. In order to get this information or to use it, ELSA performs an actual spatial reasoning as it has been defined in [3]. As a matter of fact, from poorly relevant spatial knowledge such as contour line, vegetation or ridge maps,

ELSA must infer the definition of special units of terrain called “small panels” by snow specialists and which are relevant for analysis (they are homogeneous from the analysis criteria points of view). Meanwhile this definition in small panels is not relevant enough and ELSA must also infer the properties of these small panels to perform its analysis.

These inferences are taken into account by the knowledge base system. In this paper we emphasize on terrain inference: the inference of spatial relevant properties from poor spatial knowledge. For example, here is the spatial definition of a small-panel called pp1. At the beginning of the session, this small panel is defined only by the list of triangles included in it. As it has been written in Shirka, the syntax is frame like.

```
{pp1
  is-a          = small-panel ;
  contains     = tr2 tr93 }
```

In order to make an analysis of the avalanche starting zone, ELSA needs more relevant information and, to that extent, infers a more complete description of the small panel pp1. All the fields inferred by ELSA are obtained by the use of inference methods (as presented above), particularly, pattern-matching inference and procedural attachment.

```
{pp1
  is-a          = small-panel ;
  area         = 6850. ;
  c-gravity    = %point-552 ;
  diameter     = 115. ;
  slope-%     = 68. ;
  is-in        = tende ;
  contains     = tr2 tr93 ;
  boundary-points = po4 po6 po5 po1 ;
  connected-panels = pp2 pp3 ;
  borders      = %border-589 %border-590 ;
  close-ridges = ar1 ar3 ar4 ar5 ;
  above       = pp3 }
```

Reasoning maintenance is interesting in an interactive environment for spatial reasoning. As a matter of fact, the caching of inferences is necessary because of the size of the spatial knowledge base and the amount of inferences. In ELSA, an avalanche path can easily contain more than 500 triangles and 50 small panels and ridges. Without caching the time taken for the inferences will forbid any interactive use of the system, while ELSA is dedicated to decision support and thus needs interactive use.

But, in this kind of context, the user is also supposed to modify given knowledge. In ELSA, the user can change the vegetation of a part of a small panel (in order to simulate protection works for instance), or modify the definition of a small panel (toward a more accurate decomposition of space). As a result, the spatial properties of these small panels must be re-inferred. In order to keep the base consistent, a RMS is necessary.

Although ELSA is based on Shirka/TMS, it can take advantage of the RMS in order to manage dynamicity in spatial reasoning. Fig. 2 gives a good example of interest of such a RMS.

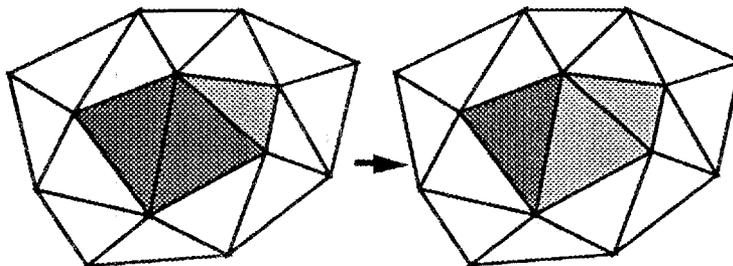


Fig. 2. In a triangulation of space, two polygons are defined through the set of triangles which are included in them. The inferences described below are made on those polygons. If a triangle changes its owner, the RMS must invalidate the cached inferences which were concerned by these two polygons. Meanwhile, the inferences conducted on the other polygons are not modified. The invalidation remains local.

As a summary, it appears that spatial reasoning applications can take advantage of classical RMS abilities. More precisely, the spatial locality can be translated in the dependency graph.

2.5. PERFORMANCE TESTS ON THE ELSA SYSTEM

Some inference times are given in order to illustrate our claims. They show how the caching is attractive and also why the RMS is useful. The tests have been performed on the same hardware as above.

Table 1. This first set of queries concerns caching; each query requires the computation of the close ridges of a panel. This second set of queries also concerns caching but queries compute the set of panels connected to a precise panel. No results about Shirka alone are provided because response times are prohibitive (in fact, from this unique test, we can conclude that ELSA is not viable without caching).

maintenance level	Shirka	Caching	RMS
Shirka: val? pp34 close-ridges	8.78	4.21	4.77
Shirka: val? pp34 close-ridges	8.72	0.	0.
Shirka: val? pp33 close-ridges	17.12	1.8	2.21
Shirka: val? pp32 close-ridges	19.01	2.11	2.49
Shirka: val? pp1 close-ridges	12.14	1.47	1.74
Shirka: val? pp2 close-ridges	8.71	1.14	1.36
Total 1	74.48	10.73	12.57
Shirka: val? pp26 connected-panels		65.32	75.92
Shirka: val? pp26 connected-panels		0.	0.
Shirka: val? pp27 connected-panels		4.9	6.7
Shirka: val? pp27 connected-panels		0.	0.
Shirka: val? pp30 connected-panels		4.15	5.37
Shirka: val? pp1 connected-panels		3.68	4.35
Shirka: val? pp2 connected-panels		3.53	4.96
Total 2		81.58	97.3

Table 2. After the tests that produced Table 1, the user changes the terrain description transferring one triangle (tr78) from a panel to another (just as in Fig. 2). The former queries are processed at new. In the first case (single caching), the user must clear the base and load it again. The time required for those operations is not taken into account.

maintenance level	Caching	RMS
Shirka: sup-val pp26 contains tr78	0.89	0.71
Shirka: aj-val pp27 contains tr78	0.1	1.81
Shirka: val? pp34 close-ridges	4.2	0.
Shirka: val? pp33 close-ridges	1.81	0.
Shirka: val? pp31 close-ridges	2.14	0.83
Shirka: val? pp1 close-ridges	1.49	0.83
Shirka: val? pp2 close-ridges	1.16	0.83
Shirka: val? pp26 connected-panels	65.87	6.16
Shirka: val? pp27 connected-panels	5.17	8.08
Shirka: val? pp30 connected-panels	4.17	7.
Shirka: val? pp1 connected-panels	3.7	5.17
Shirka: val? pp2 connected-panels	3.52	5.49
Total (initial inference + modification + re-inference)	175.8	134.21

With single caching, inference time is considerably reduced. A further discussion will give some explanations of some surprising results (especially the reduction of the *first* inference time). With the RMS, inference times are slightly increased in comparison with single caching inference times but the gain toward Shirka is obvious.

The second kind of queries shows the gain of time thanks to reasoning maintenance system. The comparison is made between single caching and RMS. The total line in Table 2 shows that the gain provided by the RMS is very important.

2.6. NEW EXPLANATIONS FOR THESE RESULTS: CONE EFFECTS

The observation made (comparing ELSA with or without RMS) are counter-intuitive at first sight:

- 1) Of course, the second call to the same inference takes no time with the RMS while, in spite of its the filtering capabilities, in Shirka, it still takes a while.
- 2) Even the first call is faster with the RMS than without (with a factor ≥ 2)!
- 3) Moreover, the time required to answer the same query against another object is reduced of a factor 8.

So these evaluations reveal a synergistic effect between inferences. These effects can be summarized as:

Backward cone effect: there is a backward cone effect when a datum is used several times in the computation of another. This can be stated in another way: the more used the datum, the better the caching. This effect is as much interesting as the datum is expensive to compute. Backward cone effect is able to explain the results above for points (2) and (3). Intermediate inferences performed use each other several times in order to obtain the high-level (or requested) data. With the RMS, these intermediate data are computed only once. For the same reason that the inferences of different data share the same intermediate inferences, after the computation of an item, the required time to answer the same query against another object is reduced. The two former points explain why the system is also faster on the re-computation after a change.

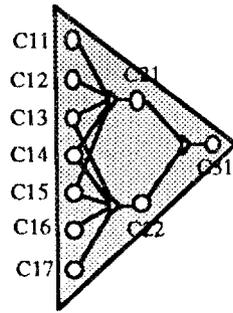


Fig. 3. In order to obtain C31, the system must infer C21 and C22 which in turn necessitates other inferences. Their computation can take advantage of caching because they share common inferences. This explains that the inferences produced with caching are faster even for their first computation.

Forward cone effect: the more used the datum, the worse the invalidation. As before, there is a forward cone effect when a datum is used for the inference of a important number of other pieces of knowledge. The forward cone effect is a negative effect, it reflects the necessary work in order to invalidate a cached result. It explains the classical results of observation (1) with Shirka/TMS.

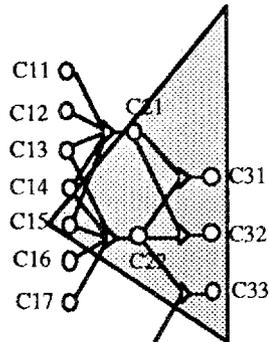


Fig. 4. The whole graph represents the inferences made by the inference engine. The shaded part of the graph is invalidated after the suppression of C15. We can see, *qualitatively*, that this shaded part looks like a “forward cone”. The larger is this cone, the less interesting is the RMS because the number of inferences to launch is nearer from the numbers of all the inferences.

The problem that will be addressed in the remaining is: how is it possible to quantify these effects? and which conclusions to draw for the use of a RMS in a particular application. It is obvious that the attraction of a RMS in an application will result in a trade off between backward and forward cone effects.

3. A SPACE OF REASONING: THE DEPENDENCY GRAPH

Here is an attempt to generalize the results we obtained with the ELSA experiments in order to state what kind of reasoning/application can benefit from a RMS.

Real efficiency of RMS is very difficult to evaluate because a lot of factors have to be taken into account: not only the number of nodes and justifications but also the way they are organized in cycles of different kind and the order of firing rules. Moreover, the performances of RMS depend heavily of the kind of use. Here, we do not address these complexity problems but the conditions under which a RMS is useful in order to maintain a reasoning. So, worst case analysis is not a suited measure of the performances of the system and an abstract computation of the algorithm complexity is not very useful. What is important for real applications is not the theoretical complexity analysis of the program used for reasoning maintenance but the real complexity of the RMS when confronted with the real reasoning. To that extent, we exhibit some results for graphs with particular restrictions that do not trigger the whole machinery of a RMS.

This section will, first, set some definitions to be used in the quantitative analysis and the restrictions used in the present study. Then the analysis is achieved for both kinds of cone effect before summarizing the results of the tradeoff between backward and forward cone effect.

3.1. NOTATION AND RESTRICTION

In order to give some precise results, some hypotheses have been done about the dependency graph. We assume that:

- H1) There is no nonmonotonic inferences. This is not an important restriction when assumed the second hypothesis. In fact, nonmonotonic inference in a graph without loops is a problem for the inference system but not for the RMS.
- H2) There is no loops in the graph. This assumption is quite restrictive. In fact, it is restrictive regarding the complexity analysis of RMS, but it is not for a lot of applications.
- H3) The analysis below only considers average values and hypothesizes the homogeneity of the graph. With regards to real application, this is the most restrictive hypothesis. The general aspect of reasoning will be evaluated and quantified on the basis of average values considered that the graph can itself be decomposed in several little sub-graphs in which it is possible to cancel or activate reasoning maintenance.

All those hypotheses are set for reason of simplicity. Of course, the quantitative analysis of reasoning for RMS have to be fulfilled with the relaxation of those hypotheses.

First, some notations have to be introduced. Let B be a knowledge base dedicated to a given application. We consider all the inferences launched all along the typical session of the application; this is called the reasoning. A particular reasoning can be represented as a dependency graph such as the one used in the RMS. If we do not care for nonmonotonic inferences (H1), it is an AND-OR graph (each inference is an and-node linking the antecedents to the consequent, each formula is an or-node linking together the possible inference of this formula).

Note that the dependency graph (as it does in RMS) does not represent the potential inference of B , but the inferences really committed. The formulas in the graph constitute the set F of formulas used in the reasoning (they can either be given by the user or inferred by the reasoning system). N is the number of all formulas in F . In F , we distinguish two sets of formulas: I is the set of initial formulas which are given and not inferred, and Q is the set of interesting formulas which are the goal of the reasoning process.

We call a chain, a sequence $f_0, j_1, f_1, \dots, j_n, f_n$ of formulas and justifications such that, for each $i \in [1, n]$, f_{i-1} is an antecedent of j_i and f_i is the consequent of j_i in the graph. n is the length of the chain (the number of justifications).

The forward depth ($df(f)$) at node f is the length of the longest chain beginning at node f (and ending at a node in Q). The backward depth ($db(f)$) at node f is the length of the longest chain ending by node f (and beginning at a node in I). Backward depth is also called the level of f .

The forward width ($wf(f)$) at node f is the number of and-node f is linked with as antecedent. The backward width ($wb(f)$) at node f is the number of and-node f is linked

with as consequent. So, $wf(f)$ and $wb(f)$ are the number of connections at front or back of an OR-node. Note that width can also be called branching factor at node f .

wf is the average number of justifications based on one formula of FQ (where $FQ = \{x; x \in F \wedge x \in Q\}$). wb is the average number of justifications of a formula of FI . In this paper, we will consider that $wb = 1$ (this means that a datum is inferred by only one way). So,

$$wf = \frac{\sum_{f \in FQ} wf(f)}{|FQ|}, \quad wb = \frac{\sum_{f \in FI} wb(f)}{|FI|}$$

$\nu(f)$ is the number of times f is used during the session, this is not the number of inferences in which it appears but the number of times these inferences are drawn. In a lot of applications these inferences are used a lot.

ρ is the ratio $\frac{|FQ| * wf}{|FI| * wb}$. Thus, here $\rho = \frac{|FQ| * wf}{|FI|}$, because of the value of wb . It is the average number of antecedents per justification in the reasoning graph.

We distinguish several constant times which are:

- τ_{inf} : the average time taken for an inference for which all the premisses are available.
- τ_{rec} : the average time taken for recording a value (result of an inference).
- τ_{dep} : the average time taken for recording a dependency (representing an inference).
- τ_{sup} : the average time taken for suppressing a dependency and a cached value.



Fig. 5. Examples of typical graphs

An additional important constant time also appears, but is not taken into account in our argumentation. It is T_{reset} , the time required for quitting and loading the application again. We guess that all these values can be easily evaluated for homogeneous reasoning. Two archetypical examples are given in Fig. 5.

Table 3. The average values of the variables for the graphs given above.

	$ FI =N$	$ Q $	$ II $	wf	wb	ρ
(1)	7	1	4	1	1	2
(2)	7	4	1	2	1	1

3.2. INFERENCE AND PROPAGATION ANALYSIS

At first sight, the time taken in order to produce the reasoning is

$$TB = \tau_{inf} * \sum_{f \in FN} \nu(f)$$

But, if the backward cone effect is accounted for, we can say (if $\rho \neq 1$) that the time taken to infer the formula f is:

$$TB(f) = \tau_{inf} * \frac{\rho^{db(f)} - 1}{\rho - 1}$$

because $\frac{\rho^{db(f)} - 1}{\rho - 1}$ is the size of a complete ρ -ary tree of depth $db(f)$. If $\rho=1$, then $TB(f) = \tau_{inf} * db(f)$. Hence, the total time for the inferences of the session, if no result of inference is recorded (and $\rho \neq 1$), is:

$$TB = \tau_{inf} * \sum_{f \in Q} \frac{\rho^{db(f)} - 1}{\rho - 1}$$

It is noteworthy that the ratio used in $TB(f)$ is the number of inferences in the backward cone. If other hypotheses are taken into account (no homogeneity, no tree structure...), the formula can be replaced in $TB(f)$ by another expressing the number of inferences in the cone. With the recording of all the inferred formulas the time is:

$$TB_{cach} = \sum_{f \in FN} (\tau_{inf} + \tau_{rec}) = |FN| * (\tau_{inf} + \tau_{rec})$$

and

$$TB_{RMS} = \sum_{f \in FN} (\tau_{inf} + \tau_{rec} + \tau_{dep}) = |FN| * (\tau_{inf} + \tau_{rec} + \tau_{dep})$$

As a result, the gain for RMS is:

$$GB = \sum_{f \in FN} [(\nu(f) - 1) * \tau_{inf} - (\tau_{rec} + \tau_{dep})]$$

the formula is the same for caching without the reference to τ_{dep} .

Table 4. The values for the graphs given as examples. Notes that they are not multiplied by the same factors. As a result, case (1) do not profit from caching (and this is true whatever is the total depth of the graph).

	TB ₁	TB ₂	TB _{cach}	TB _{RMS}	GB
(1)	3	3	3	3	0
(2)	8	8	6	6	2

3.3. INVALIDATION ANALYSIS

A RMS is useful for invalidation (otherwise, rough caching is enough as it is in forward chaining systems). Invalidation will lead to the forward cone effect. This effect is now evaluated. We consider that one given formula f in I is modified and that the user asks the same queries Q as before. Because of the caching system, the answers

recorded are no valid any more. With a single caching system, we need to re-infer all the formulas. The time required is so

$$TF_{cach} = TB_{cach} + T_{reset} + TB_{cach}$$

where $TB_{cach} = |FNI| * (\tau_{inf} + \tau_{rec})$ as above.

Another solution uses a reasoning maintenance system. In that case,

$$TF_{RMS} = TB_{RMS} + T_{inv} + T_{reinf}$$

with $TB_{RMS} = |FNI| * (\tau_{inf} + \tau_{rec} + \tau_{dep})$ as above,

and $T_{inv} = N_{inval} * \tau_{sup}$ and $T_{reinf} = N_{inval} * (\tau_{inf} + \tau_{rec} + \tau_{dep})$,

and $N_{inval} = \frac{wf^{df(f)+1} - 1}{wf - 1} - 1$ if $wf \neq 1$ and $df(f)$ otherwise, this is, again, the size of a wf-

ary tree of depth $df(f)$, so this is the size of the forward cone starting at f . The branching factor is wf because it has been considered justifications with only one consequent, otherwise, the branching factor would have been $wf * nbc_{sq}$ (in which nbc_{sq} is the average number of consequents).

As a result, the gain given by the RMS is:

$$GF = N * (\tau_{inf} + \tau_{rec}) + T_{reset} - N_{inval} * (\tau_{inf} + \tau_{rec} + \tau_{dep} + \tau_{sup})$$

If we ignore resetting time and set that $(\tau_{inf} + \tau_{rec}) * k = \tau_{inf} + \tau_{rec} + \tau_{dep} + \tau_{sup}$, the RMS must be attractive when $N_{inval} * k < N$ which must be true most of the time.

Table 5. The results of the invalidation phase for the graphs given above ($\tau_{cach} = \tau_{inf} + \tau_{rec}$ and $\tau_{RMS} = \tau_{inf} + \tau_{rec} + \tau_{dep} + \tau_{sup}$). The graphs are not big enough to illustrate interesting properties: both cases do not appeal for a RMS. In particular, locality do not appear (after each modification, an important part of the graph must be revised). It becomes more attractive if we consider 10 independent graphs as in case 2' in which the invalidation is useful.

	TF_{cach}	N_{inval}	TF_{RMS}
(1)	$6 * \tau_{cach}$	2	$6 * \tau_{RMS} + 2 * \tau_{sup}$
(2)	$12 * \tau_{cach}$	6	$12 * \tau_{RMS} + 6 * \tau_{sup}$
(2')	$120 * \tau_{cach}$	6	$66 * \tau_{RMS} + 6 * \tau_{sup}$

A pure static evaluation can be given with p modifications of data and the whole set of queries between them:

$$T(p) = p * TB$$

$$T_{cach}(p) = p * T_{reset} + (p+1) * TB_{cach}$$

$$T_{RMS}(p) = TB_{RMS} + N_{inval} * p * (\tau_{inf} + \tau_{rec} + \tau_{dep} + \tau_{sup})$$

3.4. REASONING (ABOUT/FROM) THE GRAPH

As said above, the main problem consists in evaluating the tradeoff between both cone effects. The important questions to ask for a particular application are: Can it benefit from caching? Does caching need dynamicity management? Is a RMS suited for dynamicity management or is it better to recompute everything?

It is noteworthy that these questions cannot be answered independently. Moreover, they are not directive; in particular, dynamicity management does not imply the use of a RMS. Nevertheless, reasoning dynamics must be taken into account. As a matter of fact, the performances of the system depend on the relations between query and modification time. The result will not be the same if there is a new query after each modification or if there is an important number of modifications between each query phase.

4. CONCLUSION

The problem we addressed was the evaluation of the benefits of caching and RMS in knowledge based applications. To that extent, we first show some results expected on a general purpose tool and some results obtained on a real world application. The results, at the advantage of the RMS, were not expected. We explained then by producing two informal models of the actions of caching and RMS on the reasoning: the so-called cone effects. Then, we quantified the amount of work required in order to demonstrate some facts (or resolve one problem). The equations we obtained revealed the presence of the cone in the quantifications of the number of inferences they contain.

This is a first attempt in order to characterize the usefulness of caching and RMS. It has to be continued by a better knowledge of reasoning dynamics and by relaxing the hypotheses we assumed on the graphs. Finally, the advantage of using a RMS in an application is seen as a tradeoff between both principles. More examples and experiments, together with a discussion of further and related works can be found in [5].

REFERENCES

1. F. Rechenmann and P. Uvietta, "SHIRKA: an object-centered knowledge base management system", *Artificial intelligence in numerical and symbolic simulation*, 9-23 (1991).
2. J. Euzenat, "Cache consistency in large object knowledge bases", Laboratoire ARTEMIS/Imag internal report (1990).
3. L. Buisson, "ELSA : a problem solving environment for avalanche path analysis", *Artificial intelligence in numerical and symbolic simulation*, 25-49 (1991).
4. L. Buisson, "Reasoning on space with knowledge object centered representation", *Lecture notes on computer science*, 409:325-344 (1990).
5. L. Buisson and J. Euzenat, "A quantitative analysis of reasoning for RMSes", Laboratoire ARTEMIS/Imag Internal report (1991).

A Strategy for the computation of Conditional Answers *

Robert Demolombe
ONERA/CERT
2 avenue E.Belin B.P. 4025
31055 Toulouse
France
e-mail: demolombe@tls-cs.cert.fr

June 1991

Abstract

We consider non-Horn Deductive Data Bases (DDB) represented in a First Order language without function symbols. In this context the DDB is an incomplete description of the world. A first approach to reduce the incompleteness is to add to the DDB some kind of default rules, in order to automatically assume missing information. The second approach, which is adopted in this paper, is to provide to the user the conditions which guarantee the validity of the answer. These conditional answers are generated by standard reasoning, and not by default reasoning.

Then the problem is the following : if T represents the DDB and q the query, and if there is no direct answer to q , we want to derive the more general conditions c such that : $T \vdash q \leftarrow c$. We present a strategy, GASP, designed for this purpose. It is defined by meta rules, and these meta rules can be used for a least fixpoint operator definition. We show that the GASP strategy is always more efficient than another usual strategy called GALP. Since in the case of recursive definitions the answers may be infinite GASP has been adapted into GRASP in order to only compute ground conditional answers. We show that the least fixpoint operator associated to GRASP computes the answer in a finite number of steps, even if the DDB contains recursive definitions.

*This work has been partially supported by the CEC, in the context of the Basic Research Action, called MEDLAR.

1 Introduction

Many works have been devoted to the *standard* approach of Deductive Data Bases (DDB) [1, 12, 5, 13]. In this approach a DDB is composed of two parts : a set of rules, the Intensional Data Base (IDB), which is a set of definite Horn clauses, and a set of facts, the Extensional Data Base (EDB), which is a set of ground atoms. More recently this approach has been extended to disjunctive DDB where the rules are not necessarily Horn clauses [8, 3, 6] , and facts may be ground positive clauses.

In this paper we extend disjunctive DDB to the case where EDB may contain any kind of ground clauses. But the most significant contribution is to consider a new kind of answers called *Conditional Answers*. We consider two kinds of Conditional Answers : the Intensional Conditional Answers [2], which are derived from IDB, and the Extensional Conditional Answers, which are derived from $IDB \cup EDB$. Conditional answers are another way to deal with incompleteness. Indeed the usual approach is to reduce the incompleteness with some kind of meta rule like Closed World Assumption (CWA), or Generalised Closed World Assumption (GCWA) [7], in the context of disjunctive DDB, or default rules in the context of non-monotonic reasoning [10]. In the Conditional Answer approach no assumption is added to the DDB by applying some kind of default reasoning. When there is not enough information in the DDB to answer a given query, the answer provide the less restrictive assumptions which allow to infer the query.

Let's consider for example the very simple DDB : $A \vee B \leftarrow C \wedge D$, $\neg C$, and the query : A ?

In that case we cannot provide a direct answer to the query, but we can provide the conditional answer : $A \leftarrow D \wedge \neg B$. Then the user knows that A is true under the assumptions : D and $\neg B$, and he can take the decision himself to assume or not D and $\neg B$.

In the next section is presented a general definition of conditional answers. Then we present a strategy to compute conditional answers. Its efficiency is compared with another standard strategy, and we point out the particular problem of infinite answers. In the last section we propose a modification to this strategy in order to compute extensional conditional answers in a finite number of steps.

2 General definition of Conditional Answers

We consider queries which are positive literals. This assumption does not restrict generality. Indeed, if the query is a general formula $F(x)$, we define a new predicate symbol $q(x)$, we add to the DDB the formula $Q = (q(x) \leftarrow F(x))\forall x$, and the query is represented by the positive literal $q(x)$.

EDB is a set of ground formulas. IDB is a set of formulas. We consider the theory $T = IDB \cup EDB \cup Q$, where all the formulas are represented in clausal form, and each clause is Range Restricted. Moreover, as usual in the DDB context, we consider clauses **without functional symbols**.

Definition 1 : Conditional Answer

Let q be a positive literal, the conditional answer to the query q is the set of clauses :

$$\{ q\sigma \vee c \mid T \vdash q\sigma \vee c, \text{ and } q\sigma \vee c \text{ is not a tautology, and } q\sigma \vee c \text{ is minimal wrt subsumption } \}$$

A clause d is minimal with regard to subsumption, in the context of T , if there is no clause d' derivable from T such that d' subsumes d . A clause d' subsumes a clause d if there exists a substitution σ such that : $d'\sigma \subseteq d$.

The clause c is called by Reiter and de Kleer, in [11], a minimal support for $q\sigma$. The clauses $q\sigma \vee c$ satisfying these properties are called minimal implicants.

It is important to notice that computing Conditional Answers is a new kind of problem with regard to Theorem Proving and Logic Programming. The new feature comes from the fact that an answer is neither a truth value, like in Theorem Proving, nor a set of substitutions, like in Logic Programming, but a set of clauses.

This problem is deeply related to Abductive reasoning, with some particular features due to the DDB context.

Definition 2 : Extended Conditional Answer

With the same notations we define an extended conditional answer as the set of clauses :

$$\{ q\sigma \vee c \mid T \vdash q\sigma \vee c, \text{ and } q\sigma \vee c \text{ is not a tautology, and there is no clause } c' \text{ such that } : T \vdash q\sigma \vee c' \text{ and } c' \text{ subsumes } c \}$$

We can easily see that, for a given query, the extended conditional answer contains the conditional answer. The only difference is that for clauses in the extended conditional answer there is no guarantee that c is not a theorem of T ; this means that $\neg c$ may be an inconsistent assumption.

3 Intuition of the strategy

The strategy presented in this section has been specifically designed to compute extended conditional answers, and it is based on the L-inference presented in [4]. To get a conditional answer from an extended conditional answer we have to test, for every given clause $q\sigma \vee c$ in the extended conditional answer, if c is derivable or not from T . For this purpose we can use any strategy designed for Theorem Proving.

The strategy is called GASP, an abbreviation for Generate As Soon as Possible. It is informally described in this section with a simple example. For this description we shall call *relevant theorem* for a given query, a clause derivable from T containing the query, or one of its instances.

The idea is, in a first step, to select the axioms in T which are relevant theorems. In the current step one, or several, generated relevant theorems are resolved with an axiom in T . The resolvent is a new relevant theorem which can be used in the next step. At each step tautologies and subsumed clauses are removed.

Let's consider, for example, the theory T with the axioms :

- (1) $Px \vee \neg Qx$ (2) $Px \vee \neg Rx$ (3) $Qx \vee Rx \vee \neg Sx$ (4) $Ux \vee \neg Px$
 (5) $Pc \vee \neg Uc \vee Tc$ (6) $Sa \vee \neg Ta$ (7) $Sb \vee Ub$

and the query ¹ : Px ?

The clauses generated by the GASP strategy are :

- Step 1 : (1) $Px \vee \neg Qx$ (2) $Px \vee \neg Rx$ (5) $Pc \vee \neg Uc \vee Tc$
 Step 2 : (8) $[Px \vee Rx \vee \neg Sx]$ (9) $[Px \vee Qx \vee \neg Sx]$ (10) $Px \vee \neg Sx$ (11) $[Pc \vee \neg Pc \vee Tc]$
 Step 3 : (12) $Pa \vee \neg Ta$ (13) $Pb \vee Ub$

In the Step 1 are generated the axioms in T containing an instance of the query Px . In the Step 2 a standard resolution generates (8) (resp. (9)) from (1) (resp. (2)) and (3). The clause (11) is generated from (4) and (5). An hyperresolution generates (10) from (1), (2) and (3). Notice these resolutions preserve Px , or an instance of Px , in the resolvent. At the end of Step 2 the clauses (8) and (9) are removed because they are subsumed by (10), and (11) is removed because it is a tautology. The removed clauses are represented between brackets. In the Step 3 (12) and (13) are respectively generated by resolving (6) and (7) with (10).

Though GASP generates only relevant theorems the final result, here the clauses : (1),

¹The predicate arguments are not between parenthesis to have simpler notations. For example, $P(x,a)$ is noted Pxa .

(2), (5), (10), (12), and (13), is not the conditional answer but the extended conditional answer. For instance the clause (13) is not in the conditional answer because it is subsumed by (14) Ub , which is derivable from (4) and (13).

As we said before for each clause in the extended conditional answer it would be possible to check if the condition is consistent or not with T in a further phase. For example to check if Ub or $\neg Ta$ are theorems, we could apply again the GASP strategy to the queries : $Ub?$, or $\neg Ta?$, in order to test if Ub or $\neg Ta$ are theorems of T .

4 Formal definition of the strategy

In this section the strategy is formally defined by meta rules. These rules express, at a meta level, the derivation control. It is important not to confuse the strategy used for meta rule evaluation, and the derivation strategy, at the object level, which is described by these meta rules. We use the following notations.

Meta-variables :

- q, l_i : literal variable; these variables are instantiated by literals at the object level.
- $\neg l_i$: literal variables; such a variable is instantiated by a literal which is the complement of l_i .
- c_i : clause variables; these variables are instantiated by sets of literals at the object level; this set may be the empty set.
- $l \vee l_i \vee c_i$: denotes the set of literals : $\{l\} \cup \{l_i\} \cup c_i$.
- $l \vee c_1 \vee \dots \vee c_n \vee c_0$: denotes the set of literals : $\{l\} \cup c_1 \cup \dots \cup c_n \cup c_0$.

Meta-predicates :

- $Query(l)$: this predicate means that we have to find the extended conditional answer to the query l .
- $Ax(c)$: this predicate means that c is an axiom of T .
- $Th(c)$: this predicate means that c is a theorem of T .

Definition 3 : GASP Strategy

(1) $\text{Query}(q) \wedge \text{Ax}(q \vee c) \rightarrow \text{Th}(q \vee c)$

(2) $\text{Query}(q) \wedge \text{Th}(q \vee l_1 \vee c_1) \wedge \dots \wedge \text{Th}(q \vee l_n \vee c_n) \wedge \text{Ax}(\neg l_1 \vee \dots \vee \neg l_n \vee c_0)$
 $\rightarrow \text{Th}(q \vee c_1 \vee \dots \vee c_n \vee c_0)$

One could notice that it is not usual to have dots in a formal definition. We have used dots here to make the definition more easy to read, however it would not be difficult to replace the dots by recursive definitions without dots.

We define a meta theory MT containing the rules (1) and (2), the sentence $\text{Ax}(c)$, for each clause c in T , and the sentence $\text{Query}(q)$, where q is the literal denoting the initial query.

The meta rules are evaluated with a trivial strategy which is an incremental saturation by level, with elimination of subsumed clauses and tautologies. Here incremental means that when a new sentence is generated by a meta rule, at least one of its premisses in the meta rule has to be a new sentence in the computation of the previous level.

The sets of sentences generated by saturation by level are denoted by : $S_0, S_1, \dots, S_i, \dots$

S_0 contains all the sentences derivable in one step by the rules (1) and (2) from MT. A sentence is derivable by a rule if there exist a rule instance whose consequence is this sentence, and all its premisses are satisfied by MT. All the tautologies, and all the sentences subsumed by a sentence in MT or S_0 are removed from S_0 . We call ΔS_0 the resulting set of derived sentences.

We define S_{i+1} and ΔS_{i+1} in function of S_i and ΔS_i in the following way. We consider all the sentences derivable using the rules (1) and (2) from MT and S_i and we remove from this set all the tautologies and all the sentences subsumed by a sentence in MT or S_i . The resulting set of sentences is called ΔS_{i+1} . Then S_{i+1} is defined by : $S_{i+1} = S_i \cup \Delta S_{i+1}$.

If M is any meta predicate, we say that the sentence $M(c')$ subsumes the sentence $M(c)$ iff the clause c' subsumes the clause c . We also say that $M(c)$ is a tautology if c is tautology.

We say that the premisses of the rule (2) (a similar definition applies to rule (1)) are satisfied by a set of sentence S iff :

- the following set of sentence is in S , or S contains sentences whose some factors are : $\text{Query}(Q), \text{Th}(Q_1 \vee L_1 \vee C_1), \dots, \text{Th}(Q_n \vee L_n \vee C_n), \text{Ax}(\neg L'_1 \vee \dots \vee \neg L'_n \vee C_0)$; where $Q, Q_1, L_1, C_1, \dots, Q_n, L_n, C_n, C_0$ are literals or clauses at the object level,
- there exists a most general unifier σ which is solution of the equations :

$$Q = Q_1 = \dots = Q_n \quad L_1 = L'_1 \quad L_2 = L'_2 \quad \dots \quad L_n = L'_n.$$

In that case the instantiation of the meta variables is :

$$q = Q\sigma \quad l_i = L_i\sigma \quad \neg l_i = \neg L'_i\sigma \quad c_i = C_i\sigma$$

and the generated sentence is $\text{Th}(q \vee c_1 \vee \dots \vee c_n \vee c_0)$.

The equations $L=L'$, where L and L' denote $P(t_1, \dots, t_p)$ and $P(t'_1, \dots, t'_p)$, or $\neg P(t_1, \dots, t_p)$ and $\neg P(t'_1, \dots, t'_p)$, are short hands for the set of equations :

$$t_1 = t'_1 \quad t_2 = t'_2 \quad \dots \quad t_p = t'_p$$

It is easy to show that the interpretation we have defined for the meta rules defining GASP provides a definition for a least fixpoint operator.

5 Comparison with other strategies

It is interesting to compare GASP with another very intuitive strategy based on the idea of the decomposition of problems into sub-problems “à la Prolog”. Here the problem is to compute the extended conditional answer to a query : $A?$. If there is an axiom containing A in the theory T of the form : $A \vee \neg B_1 \vee \dots \vee \neg B_i \vee \dots \vee \neg B_n$, where the B_i s may be positive or negative literals, we can generate sub-problems, i.e. new queries, of the form : $B_1?, B_2?, \dots, B_i?, \dots, B_n?$. Indeed we know that any answer to a query like $B_i?$ is of the form : $B_i \vee c_i$, and therefore any set of answers can be resolved by an hyperresolution with the axiom to generate new answers of the form : $A \vee c_1 \vee \dots \vee c_i \vee \neg B_{i+1} \vee \dots \vee \neg B_n$. This strategy is called GALP, which is an abbreviation for Generate As Late as Possible. It can be defined by meta rules in the same style as for GASP.

Definition 4 : GALP Strategy

$$(1) \text{Query}(q) \wedge \text{Ax}(q \vee c) \rightarrow \text{Th}(q \vee c)$$

For each i in $[1, p]$:

$$(2.i) \text{Query}(l) \wedge \text{Ax}(l \vee \neg l_1 \vee \dots \vee \neg l_i \vee \dots \vee \neg l_p) \rightarrow \text{Query}(l_i)$$

Endfor;

$$(2) \text{Query}(l) \wedge \text{Th}(l_1 \vee c_1) \wedge \dots \wedge \text{Th}(l_n \vee c_n) \wedge \text{Ax}(l \vee \neg l_1 \vee \dots \vee \neg l_n \vee c_0) \\ \rightarrow \text{Th}(l \vee c_1 \vee \dots \vee c_n \vee c_0)$$

In the particular case where we impose the l_i s to be positive literals, and c_0, c_1, \dots, c_n denote the empty clause, the GALP strategy is very close to strategies like : Magic Sets [1], ALEXANDER [12], QSQ [13], or APEX [5]. In that case the axioms in (2.i) are Definite Horn clauses, and the generated theorems are ground atoms. The only difference is that in these strategies the answers to sub-queries are not computed in parallel. GALP can be easily adapted in order to impose to compute the sub-queries in sequence, as we did in [3].

Unfortunately it can be shown that in every cases GALP generates a superset of the clauses generated by GASP, and then it is always less efficient. However, as it is noticed in [4], GASP may generate an infinite set of clauses when the initial theory contains recursive definitions. Nevertheless if we are interested in Extensional Conditional Answers containing only ground clauses, it is possible to adapt GASP in order to prevent infinite derivations. That is the purpose of the next section.

6 Strategy for Extensional Conditional Answers

The adapted strategy is based on the following interesting property of Range Restricted clauses : if a ground clause is derivable, by Resolution, from a set of Range Restricted clauses, and if s is the composition of all the most general unifiers used in the proof of that clause, then, if we apply s to any clause in the proof, we get a ground clause. The idea is to design a strategy, based on this property, which generates only “ground proof trees”, i.e. proof trees where all the clauses are ground clauses. The intuition of the strategy can be presented with the following example :

- (1) $Lxy \vee \neg Pxy \vee \neg P'xy$ (3) $Pay \vee \neg Ty$
 (2) $Lxy \vee \neg Rxy \vee \neg R'xy$ (4) $Rxb \vee \neg Ux$
 (5) $P'xy \vee R'xy \vee \neg Sxy$

Let's consider the corresponding connection graph, as defined by Naqvi and Henschen in [9] (see Figure 1). In this graph the nodes are clauses, and we can imagine these clauses as active agents able to send or to receive queries or answers, and able to store the answers. The role of these queries and answers is to find the unifiers of the proofs whose composition can lead to ground unifiers. They are of a different sort than the initial query and the conditional answers.

For example if the clause (1) receives the query $Lxy?$, then it sends the two queries $Pxy?$ and $P'xy?$ along the edges starting from the clause (1).

When the clause (1) sends to the clause (3) the query $Pxy?$, the meaning of this query is : “what would be the most general unifier if the clause (1) would be resolved with the clause (3) on the literal Pxy ?”. The returned answer in that case is $Pay!$.

We can also imagine that each clause can store the answers it has received from the other clauses, and that these stored answers can be used to generate new answers. For example the clause (1) can use the answer Pay! received from (3) to send the answer Lay! to the query Lxy? .

We can easily see that after receiving the query P'xy? the clause (5) sends to the clause (2) the query $\neg\text{R'xy?}$, and the clause (2) sends to the clause (4) the query Rxy? . The clause (4) returns to the clause (2) the answer Rxb! , and the clause (2) returns to the clause(5) the answer $\neg\text{R'xb!}$. Then the clause (5) returns to the clause (1) the answer P'xb! .

At this stage the clause (1) knows, from the answers Pay! and P'xb! , that there exists a proof, involving the clause (1), where the composition of all the most general unifiers defined in that proof transforms the clause (1) into a ground clause. Then the clause (1) can generate the corresponding ground instance : (6) $\text{Lab} \vee \neg\text{Pab} \vee \neg\text{P'ab}$, which is a ground conditional answer to the initial query.

In the further steps the clause (6) can be resolved, according to GASP strategy, with (3) to generate : (7) $\text{Lab} \vee \neg\text{Tb} \vee \neg\text{P'ab}$, or with (5) to generate : (8) $\text{Lab} \vee \neg\text{Pab} \vee \text{R'ab} \vee \neg\text{Sab}$, or with both (3) and (5) to generate : (9) $\text{Lab} \vee \neg\text{Tb} \vee \text{Rab} \vee \neg\text{Sab}$.

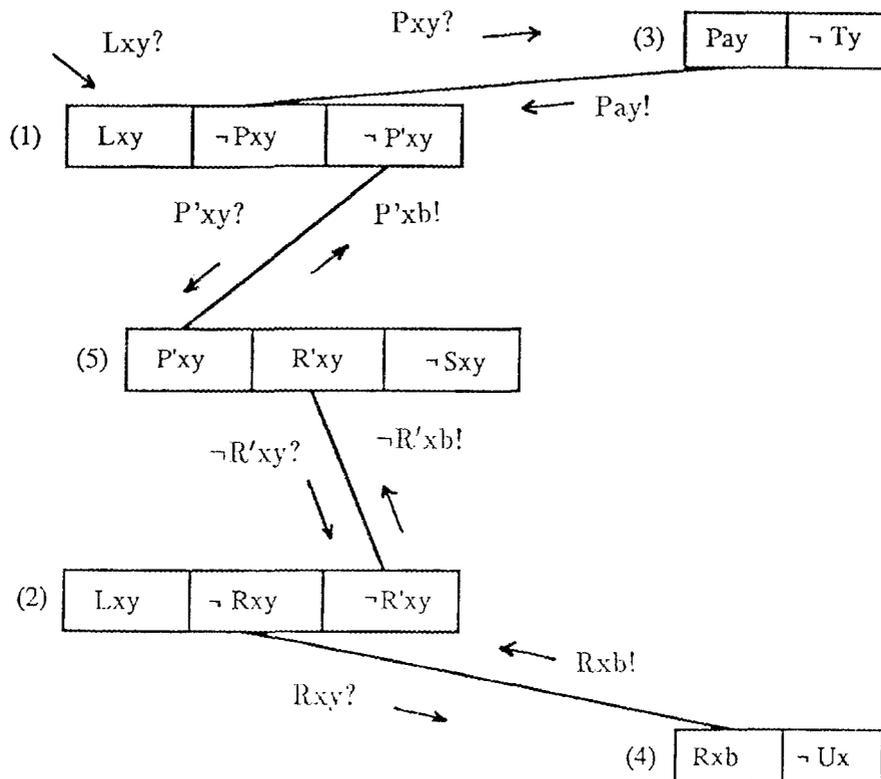


Figure 1: Connection Graph

The strategy we have informally presented is called : GRASP, for “Generate gRound As Soon As Possible”. Its description in terms of meta-rules is presented in the Definition 5,

where we use the notations :

$\text{GrAx}(c)$: c is a ground instance of an axiom.

$l?$: we have to find the l instances which appear in some derived clause of the form $l \vee c$.

$l!$: there exists a derived clause of the form $l \vee c$.

Definition 5 : GRASP Strategy

(1) $\text{Query}(q) \rightarrow q?$

For each i in $[1, n]$:

(2.i) $l? \wedge \text{Ax}(l \vee l_1 \vee \dots \vee l_i \vee \dots \vee l_n) \rightarrow \neg l_i?$

Endfor;

(3) $l_i? \wedge \neg l_{j_1}! \wedge \dots \wedge \neg l_{j_p}! \wedge \text{Ax}(l_1 \vee \dots \vee l_n) \rightarrow l_i!$

(4) $\text{Query}(q) \wedge \neg l_{j_1}! \wedge \dots \wedge \neg l_{j_p}! \wedge \text{GrAx}(q \vee l_1 \vee \dots \vee l_n) \rightarrow \text{Th}(q \vee l_1 \vee \dots \vee l_n)$

(5) $\text{Query}(q) \wedge \text{Th}(q \vee l_1 \vee c_1) \wedge \dots \wedge \text{Th}(q \vee l_n \vee c_n) \wedge l_{j_1}! \wedge \dots \wedge l_{j_p}! \wedge \text{GrAx}(\neg l_1 \vee \dots \vee \neg l_n \vee c_0) \rightarrow \text{Th}(q \vee c_1 \vee \dots \vee c_n \vee c_0)$

where i, j_1, \dots, j_p are in $[1, n]$.

In this definition the meta-rules (2.i) generate sub-queries in the same way as GALP does. An important difference is that the answers to queries of the form $l?$ are not clauses but literals like $l!$. That is the reason why the computation always stops. Indeed, even if there are recursive definitions the number of answers, up to the variable names, is finite when we do not have function symbols.

The meta-rules (4) and (5) are very similar to the corresponding ones in the GASP definition, and the meta-rule (3) generates the solutions of the form $l!$.

The evaluation of these rules, with the technique defined for GASP, generates only ground theorems which are ground conditional answers.

7 Conclusion

We have presented a strategy (GASP) to generate conditional answers in the context of a non-Horn Deductive Data Base. The basic idea is to focus the derivation process on clauses which are relevant for the query.

We have compared this strategy (GASP) with an another strategy (GALP) similar to standard strategies used in the context of Deductive Data Bases for Horn clauses, and we have shown that GASP is always more efficient than GALP. GASP is defined by meta rules, and a least fixpoint operator can be associated to these rules. This computation technique prevents to repeat several times the same computation. This is a significant benefit with respect to computation techniques “à la Prolog”, or based on SL-resolution.

In the case of recursive definitions the answer may be infinite. For this particular case we have designed the GRASP strategy, an adaptation of GASP in order to derive only ground clauses. The associated least fixpoint operator always compute the answer in a finite number of steps. However at this time we have no result about the completeness of GRASP, because we have no denotational definition of what is computed by GRASP. That needs more investigations, and as to be considered as a work in progress.

It should also be clear that the definitions of these strategies have to be considered as a general framework for further refinements. Indeed there are many open choices to implement these strategies, and, depending on these choices, the performances can be strongly improved.

Acknowledgements : Many thanks to Luis Fariñas del Cerro for all our fruitfull and stimulating discussions.

References

- [1] F. Bancilhon and R. Ramakrishnan. An amateur’s introduction to recursive query processing strategies. In *Proc. ACM PODS*, 1986.
- [2] L. Cholvy and R. Demolombe. Querying a Rule Base. In *Proc. of 1st Int. Conf. on Expert Database Systems*, 1986.
- [3] R. Demolombe. An efficient evaluation strategy for Non-Horn Deductive Data Bases. In *IFIP Congress’89, extended version in : Journal of Theoretical Computer Science Num 78*. Elsevier, 1989.
- [4] R. Demolombe and L. Fariñas del Cerro. Efficient representation of incomplete information about structured objects. In J.Schmidt and C.Thanos, editors, *Foundations of Knowledge Bases Management*. Springer-Verlag, 1990.

- [5] E. Lozinski. Evaluating queries in deductive databases by generating. In *Proc of IJCAI*, 1985.
- [6] E. Lozinski. Computing facts in non-horn deductive systems. In *Proc of VLDB*, 1988.
- [7] J. Minker. On indefinite databases and the closed world assumption. In *Proc. of 6th Conference on Automated Reasoning*, 1982.
- [8] J. Minker and A. Rajasekar. Procedural interpretation of non-horn logic programs. In *Proc. Conference on Automated Deduction*, 1988.
- [9] S. Naqvi and L. Henshen. Recursive query answering with non-horn clauses. In *Proc. Conference on Automated Deduction*, 1988.
- [10] R. Reiter. Nonmonotonic reasoning. In *Annual Reviews of Computer Science*, 2, 1987.
- [11] R. Reiter and de Kleer. Foundations of assumption-based truth maintenance system. In *AAAI-87*, 1987.
- [12] J. Rohmer, R. Lescoeur, and J-M. Kerisit. The alexander method : a technique for the processing of recursive axioms in deductive databases. *New Generation Computing*, Vol. 4(Num. 3), 1986.
- [13] L. Vieille. Recursive axioms in deductive databases : the query-sub-query approach. In L.Kerschberg, editor, *Proc. 1st Int. Conf. on Expert Database Systems*. Benjamin/Cummings Pub. Comp., 1987.

INTEGRATED REASONING THROUGH ASSOCIATIVE RETRIEVAL

Thomas C. Eskridge

Computer Science Department &
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003-0001, USA
(teskridg@nmsu.edu)

ABSTRACT

Deduction, induction and analogy have traditionally been treated as separate processes each requiring specialized machinery. We present a hybrid connectionist - symbolic approach that seamlessly integrates these forms of reasoning by way of associative retrieval.

1. INTRODUCTION

Traditional research in machine learning has taken a componential view of reasoning where deduction, induction, and analogy are studied separately with a different computational mechanism proposed for each. While this approach has value in identifying key issues for each technique, it also has problems in that the research generally does not make any attempt to integrate the techniques in an overall cognitive architecture.

We have designed and implemented a computational model in which small variations on a single mechanism, *associative retrieval*, can perform deductive, inductive and analogical reasoning. Similar notions of using a uniform mechanism to perform the three reasoning tasks have been proposed [1,2,3]. Our notion differs from these in that we employ the principles embodied in the Continuous Analogical Reasoning theory to constrain and focus what is retrieved, ensuring the retrieval of the most relevant, useful information available [4,5].

This paper begins with a discussion of Continuous Analogical Reasoning, motivating the need for interactions between the stages of analogy and comparing it to Discrete Analogical Reasoning. It then describes the hybrid symbolic-connectionist knowledge representation and processing mechanisms of the ASTRA program. It is the unique combination of structure and processing in ASTRA which allows the complex, continuous interactions to take place among all stages of the analogical reasoning process. The system's behavior under different forms of reasoning is then discussed, which shows success in achieving integrated reasoning and provides impetus for future research.

2. ASTRA: AN OVERVIEW

Analogical reasoning is typically divided into three stages: retrieval, mapping, and evaluation & use. The retrieval stage involves accessing knowledge from long-term memory (called the source) that can be applied to the current problem. In the mapping stage, the objects and relations of the source are placed into correspondence (also called a mapping) with the objects and relations in the target. By extension of the mapping, the evaluation &

use stage takes knowledge present in the source but not in the target (conjectures) and introduces them into the target domain. The conjectures are transferred to the target by replacing objects from the source with their corresponding objects in the target, and asserting the modified conjecture in the target. The evaluation & use stage then evaluates the new knowledge, checking to see if the current goal has been met, and setting new goals for the system.

ASTRA is a computational model of human analogical reasoning developed to encompass the entire range of analogical reasoning, rather than an isolated phenomenon [6,7]. As prescribed by the Continuous Analogical Reasoning theory, ASTRA models the three stages mentioned above *and* their interactions. The interactions between stages can be considered as soft constraints or “preferences” [8] which modify the processing done in each stage to reduce the search space and focus reasoning on relevant information. The interactions make the analogical reasoning process more efficient by reducing the search space, as well as more robust by focussing reasoning on relevant information.

Discrete Analogical Reasoning systems are those systems that do not promote interaction among the stages of analogy. These systems generally model only one stage of the analogical reasoning process. One justification is that the researcher is only interested in one of the stages, thus modeling only it. Another justification of this approach is that it is easier to implement one component at a time, with the idea that, once components are developed for each stage of analogy they can be tied together to create a complete system. I argue that modeling a single stage of the analogical reasoning process will be inadequate for two reasons: 1) The modeled portion will also include mechanisms to perform processing that would normally be done by another stage. An example of this is the mechanism that creates an initial mapping when modelling only the mapping stage. This information would normally be created by the analog retrieval process, where the source is examined with respect to features of the target to determine its relevance to the current situation. Thus, to adequately model just a single stage, the interactions with the other stages must be taken into account; 2) When combined into a complete system, the discrete approach will lack the efficiency of the continuous approach because it discards search constraining information generated by the stages instead of making it available to the other stages. As in the previous example, the search for a source analogy will necessarily involve the comparison of the target to prospective sources. Generally, the source that shares the most features with the target will be selected. If the correspondence information is not passed on to the mapping stage, the mapping stage must reproduce portions of the search done previously by the retrieval process in order to find the initial mapping. As in many cases in computer science, the lack of efficiency of an algorithm can result in lack of capability as well. Thus, I argue that from a psychological standpoint, it is more difficult to determine what parts of a discrete modelled stage are actually part of the stage and what parts are required due to the lack of interaction. From a computer science standpoint, the redundant search of required by a discrete analogical reasoning system can result in a limited ability to handle scaled-up problems. Our hypothesis is that a better understanding of analogy will result from looking at analogy as a continuous process rather than as a set of discrete components.

The premise of this paper is that a cognitive architecture can be built based on the continuous interaction mechanisms in ASTRA that makes no procedural distinctions between deduction, induction and analogy. Instead, the style of reasoning performed depends entirely on the type of information retrieved during the retrieval process. If the information is a rule, the mapping stage unifies the antecedent and instantiates the consequent. The evaluation & use stage transfers the consequent to the target and starts a new retrieval on the enhanced target. If the information retrieved is a set of sources, then a generalization process may be invoked to induce a description of the set. This description can then be used as the source from which to transfer information to the target. If the information retrieved by the retrieval process is a single source, analogy can proceed as usual.

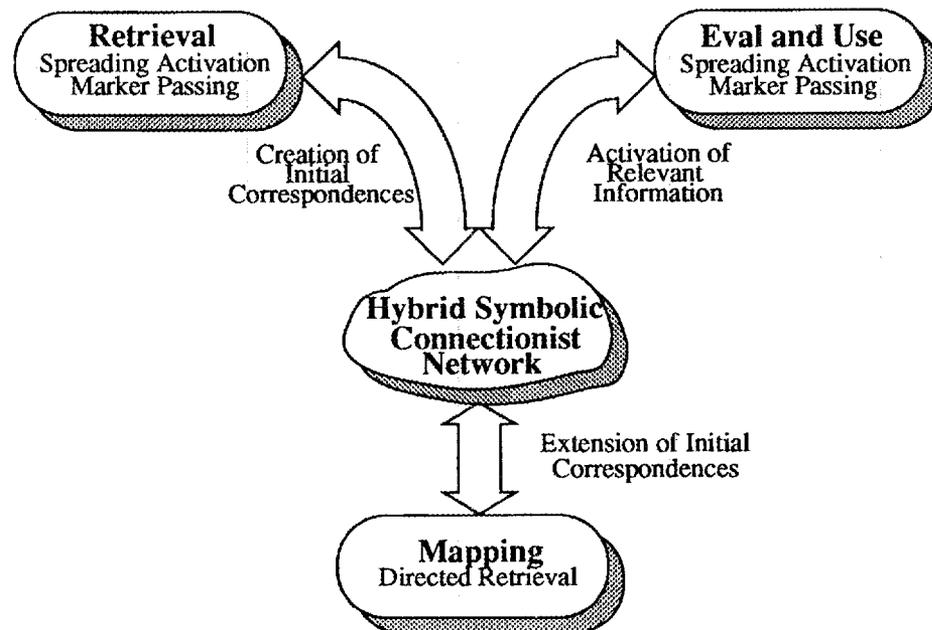


Fig. 1. Architecture of ASTRA. Like a blackboard system, all interactions between processes occurs in the hybrid symbolic-connectionist network.

3.0 ASTRA ARCHITECTURE AND PROCESSES

The architecture of the ASTRA system is shown in Fig. 1. This figure shows the bidirectional communication between each of the three processes mediated by the hybrid symbolic-connectionist knowledge representation network. The processes communicate by varying activation levels on nodes relevant to the current task. The principal mechanism for this is spreading activation. One of the difficulties in dealing with activation alone is the credit assignment problem: What nodes most significantly influenced a highly active node? To overcome this, spreading activation is augmented by a marker passing scheme which deposits on each node a pointer to the source of the activation.

The task of the retrieval process is to activate a set of sources which are semantically similar to the target problem, creating a set of initial correspondences in the process. The mapping stage interacts with retrieval by directed activation and marker passing, pressuring the retrieval stage towards analogs which are syntactically and systematically similar to the target. The evaluation & use stage spreads activation from the goal or context related aspects of the target description, pressuring retrieval towards sources that are pragmatically relevant to the target goals.

The task of the mapping process is to extend the initial correspondences to unmapped nodes in the source and target. The extension is done using the initial correspondences from the retrieval stage, pragmatic constraints from the evaluation & use stage, and systematicity principles to constrain the possible matches in the target. Nodes in the source that have no mapping after extension are considered to be conjectures and are marked for transfer by the evaluation & use stage.

The evaluation & use stage is responsible for exerting pragmatic goal and context-related pressures on retrieval and mapping. The symbolic procedures for the creation of new

nodes and links resulting from the transfer of a conjecture reside in the evaluation & use stage. Procedures for determining if current goals are satisfied and the generation of new goals also reside here. Retrieval influences this stage by suggesting new goals to pursue based on previous experiences and by suggesting different evaluation contexts for the analogy based on the type of source retrieved. The context for evaluation of the analogy will change depending on if the source analog is described in behavioral, causal, componential, or other terms.

Before presenting the details of the system, it would be useful to look at the process in overview. At the start of problem solving activities, the initial target analog description or representation is presented to the system, starting the retrieval process. The retrieval process uses a combination of marker passing and spreading of activation to both search the knowledge base for a suitable source analog and to elaborate the target analog description with deductive pattern completion inferences. Goal and context related information, if present in the initial target analog description is activated and used as a source of activation spread by the evaluation & use process. The mapping process is awakened by markers from the target analog touching nodes in other experience descriptions. Since the markers passed in the network reference the origin of the marker, the mapping process is primed with a partial mapping between the target and the potential source when it is awakened. Extension of the mapping is done for only the most highly active analogs, or any one marked by the evaluation & use's marker spreading process.

The following section describes the hybrid knowledge representation scheme and the mix of connectionist and symbolic processing used in ASTRA. There are both theoretical and practical benefits of using a hybrid representation and processing scheme. Hybrid systems are theoretically interesting simply because they are new and there is not much information available concerning their capabilities. This work can be considered to be an empirical study of the capabilities of hybrid systems for performing high-level integrated reasoning. The other benefits are purely practical: symbolic and connectionist systems have different strengths and weaknesses that are orthogonal and complementary. I believe that more powerful systems can be built with properties not present in either paradigm alone using each technique where natural and appropriate.

3.1 KNOWLEDGE REPRESENTATION

The knowledge representation used in ASTRA is a tightly-coupled integration of symbolic semantic networks with unweighted, labeled links, and sub-symbolic connectionist networks with unlabeled, weighted links. Each knowledge-level concept in ASTRA is rep-

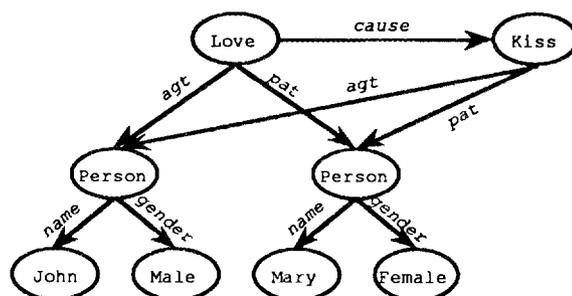


Fig. 2. Graphic representation of "Because John loves Mary, he kisses her". All links are bi-directional, with reverse links not shown.

represented by a labeled node. These nodes are shared by both the symbolic and sub-symbolic portions of the knowledge representation. Each node also has associated with it a set of microfeatures which describe the concept [9]. For example, the concept "LOVE" has {adore, respect, honor, dote, fancy, desire, favor} as its set of microfeatures. The concept "LIKE" has {adore, fancy, esteem, favor} as its set of microfeatures. The similarity between these two concepts are represented in the overlap between their microfeature sets. The similarity is determined by the activation spread from one node to the other. Thus, the more microfeatures shared between concepts, the more similar they will be.

Abstracting from the implementation level, the knowledge representation can be divided into four parts: The hierarchy of concepts, the story representation, rules, and the process-created data structures. The hierarchy defines the concepts that are known by the system and how they are related to one another by ancestry and packaging. The story representation ties together instances of concepts found in the hierarchy with relational information, forming a conceptual representation of the actors and actions of a story. Fig. 2 shows graphically the representation for the short story "Because John loves Mary, he kisses her." Rules are represented in much the same manner as stories: concepts in the anteced-

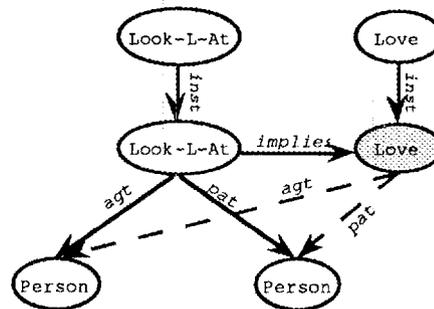


Fig. 3. Rule link "IMPLIES" in rule definition. The special "AND" node is not shown for clarity. Shaded "LOVE" node is created on rule application. Dashed links show connection of arguments to created node.

ent of a rule are tied together by a special "AND" node which is connected to the consequent nodes by an "IMPLIES" link. Fig. 3 shows an example of a simple rule. The process-created data structures are structures created in response to the actions taken by the retrieval, mapping, and evaluation & use processes. There are two such data structures: markers and bridges. A marker is simply a pointer to the node originating the mark. A bridge is a special type of link that has pointers to other bridges that it uses or that use it. The functionality of these data structures will be discussed in the next section.

3.2 PROCESSING MECHANISMS

The continuous analogical reasoning theory places some special processing requirements for a computational implementation. The greatest difficulty lies in the need for communication among the three stages during processing. There are four basic processing mechanisms used in ASTRA to implement the continuous analogical reasoning process: automatic spreading of activation, marker passing, directed retrieval, and traditional symbolic procedures. Automatic spreading of activation passes activation from a node to all neighbouring nodes [10]. A user specified decay rate attenuates the activation as it is spread in the network. The default activation rule is:

$$a_n(t+1) = \sum_i w_{in} o_i(t) + a_n(t)(1 - \Theta)$$

where $a_n(t)$ is the activation of node n at time t , w_{in} is the incoming weight from node i to node n , $o_i(t)$ is the output of node i at cycle t , and Θ is the activation decay rate. The output function of a node i is:

$$o_i(t) = \begin{cases} \frac{a_i(t)}{|\alpha_i|} & a_i(t) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where $|\alpha_i|$ is the number of links connected to node i , and θ is the activation threshold cutoff.

<i>Spreading Activation Rules</i>
SAR-1: If the link is an "INST-OF" link connecting an individual to a general concept in the hierarchy, do not decay.
SAR-2: If the link is marked as special because of the goals or context of the problem solver, increase activation by specified amount.
SAR-3: If the link's head-node has the same label as the origin of the activation, then double the amount of activation on that node.
<i>Marker Passing Rules</i>
MPR-1: If a node receives activation above the cutoff threshold, place on it a copy of the marker given to the node passing the activation.
MPR-2: If a marker is placed on a node from a different story representation, start a symbolic process that will build a bridge if necessary.

Fig. 4. Spreading Activation and Marker Passing Rules. A set of transition rules define exceptions to the default activation transfer equation. Two simple rules define the marker passing procedure.

The marker passing mechanism follows the activation spreading process and puts markers containing information about the origin of the activation on each node receiving activation. When a marker is placed on a node from a different story representation, a bridge link can be created between it and the target node. The directed retrieval process complements the above two processes in that it allows activation and markers to be spread between nodes only over specified links. The symbolic procedures used include procedures for creating bridge links and for determining untouched portions of story representations. More detail on the processing mechanisms can be found in [7].

4. REASONING IN ASTRA

In this section we describe the reasoning methods ASTRA can produce. To do this we present the discussion in terms of an example from our test domain of interpersonal relationships. For this example, ASTRA has four stories represented in memory from which to draw information. The stories are:

- 1: "Because John loves Mary, he kisses her" (shown in Fig. 2),
- 2: "Because Carol dislikes Ted, she slaps him",
- 3: "Because Jenny loves Tom, she kisses him", and
- 4: "Because Fred loves Wilma, he kisses her"

and the rule:

R1: "If a person looks longingly at another person, the first person loves the second person"

Given the target "Romeo looks longingly at Juliet" shown in Fig. 5, what predictions can be made from this?

4.1 DEDUCTION IN ASTRA

Automatic spreading of activation is begun when the target is presented to ASTRA. The target nodes "LOOK-L-AT", "PERSON(1)" (the *agt* node), "ROMEO", "PERSON(2)" (the *pat* node), "JULIET", "MALE", and "FEMALE" act as sources from which activation is spread. A marker is created for each of the target nodes and is passed along with the activation, continuing until the activation level falls below a user-specified threshold. Activation from each of the target nodes moves up into the hierarchy and back down to nodes in the same class, but in different story representations. At the same time, the evaluation & use process uses a directed retrieval process to increase the activation of any node that passes over a "CAUSE" link. The evaluation & use stage emphasizes "CAUSE" links since the goal of the system is to answer "What will happen because Romeo looks longingly at Juliet?"

Activation spreads from each activation source to all connected nodes. Activation from "LOOK-L-AT" spreads up the hierarchy and back down to node "LOOK-L-AT" where rule R1 is connected. When activation reaches the rule, a symbolic process is triggered that creates a bridge link between the "LOOK-L-AT" in the target and the "LOOK-L-AT" in the rule. Activation also travels from the target nodes across microfeatures to other nodes, and

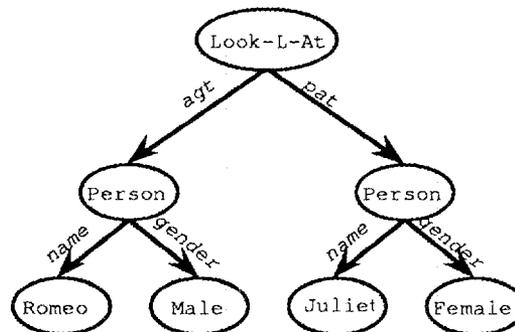


Fig. 5. Target representation of "Romeo looks longingly at Juliet". The goal is to determine what will come of this situation.

bridges are created for them also. A strength is associated with each bridge that corresponds to the degree of evidence for that correspondence. The bridge strength is the amount of activation at the bridge's head node that can be attributed to the node at the tail of the bridge. Since the target "LOOK-L-AT" and its correspondent in R1 have identical labels, the activation at "LOOK-L-AT" in R1 and the bridge is doubled by rule SAR-3. Thus, it has the highest activation of all bridges emanating from "LOOK-L-AT" in the target. The other nodes in the target also spread activation and set up bridges with corresponding nodes in the knowledge base.

Once all activation has fallen below a user specified threshold, a symbolic process is invoked to which selects the source analog from the group of competing analogs. The selection is made by choosing the maximum of the summed activation of all nodes in the story representation, combined with the activation along the bridges between the target and the story representation. In this example, rule R1 has the highest activation and is chosen as the source. The mapping process enforces systematicity in determining which set of bridges created by retrieval will yield the best mapping. This is done by preferring bridges that relate identical case relations between head and tail nodes of the bridge with the highest activation. By focussing on the bridges with high activation, the mapping process pairs the target with rule R1. In this pairing, the rule node "LOVES" is left unmapped, and so is marked as a potential node for transfer.

Evaluation & use notices that there is a node marked for transfer and sets up a symbolic process to create a new instance of that node, and create links to the corresponding nodes in the target. The evaluation & use process checks to see if what has been transferred to the target is enough to solve the current goal "What will happen because Romeo looks longingly at Juliet?" In this case, the information transferred does not provide a solution to the goal, but rather, the deduction provides a means for restructuring the target situation into an new problem, "What will happen because Romeo loves Juliet?" Evaluation & use sets this problem up as the new goal for the system.

4.2 INDUCTION IN ASTRA

Armed with a new target situation, ASTRA restarts the retrieval process. Activation spreads from the deduced target node "LOVES" through the hierarchy to the "LOVES" nodes in sources 1, 3 and 4. Activation will also go to the "DISLIKES" node of source 2 if the activation level does not fall below threshold after climbing an extra step up in the hierarchy to reach a common generalization. Once activation does reach a "LOVES" node in the source stories, the mapping process creates a bridge between the "LOVES" node in the source and the "LOVES" node in the target. The strength of this link will be high relative to the strength of a bridge between "DISLIKES" and "LOVES", because of their close proximity in the concept hierarchy. At the point where activation from the target is spread from the "LOVES" nodes in sources 1, 3 and 4, the activation is increased by the evaluation & use process because of the "CAUSE" links connecting the nodes.

Target nodes spreading activation now have new paths to explore along the newly created bridges. Activation levels that would normally not be strong enough to go through the hierarchy to other sources can now go there directly by way of the bridges. When a node sends activation along a bridge, another directed retrieval process is created by the mapping stage that checks if the relationship between the origin of the activation and the head node of the bridge is present on the tail-node. If it is, then a new bridge is set up between those nodes. Thus when "PERSON(1)" passes activation and its marker along the bridge between the target "LOVES" and the "LOVES" in 1, a similar relationship is found between the target "PERSON(1)" and the source "PERSON(1)" nodes. A new bridge is set up to reflect this re-

relationship. The new bridge is also connected to the bridge traversed as a component of that bridge and increments its strength by the amount of activation in the new bridge. The bridges then form a network of competing mapping hypotheses. In this example, the bridges that map the target to sources 1 and 4 have greater strength than those mapping the target to source 3. Source 3 requires a mapping of "MALE" \Rightarrow "FEMALE" and "FEMALE" \Rightarrow "MALE" which will have lower strength than the identical mapping of the bridges to sources 1 and 4. Thus, sources 1 and 4 will have the greatest activation and will be selected as a set of sources.

When the retrieval process proposes a set of sources instead of a rule or a single source, an induction process is spawned to produce a generalized story which encompasses the stories in the set. The induction process is done by simple hierarchy climbing and replacing differing constants by variables. In this case, the generalization procedure will produce "When a male person with any name loves a female person with any name, he will kiss her". The new structure is placed into memory with appropriate links to sources in the set. The bridges from the target to the different sources in the set are similarly generalized. The generalized bridges are connected to the target and the generalization. The strength of each generalized bridge will be the sum of the strengths of the member bridges between the target and the sources in the set. Thus, the generalization will now have the highest activation of the story representations in memory.

4.3 ANALOGY IN ASTRA

Now that the generalized story created by the induction process has the highest activation of any story representation linked to the target, the mapping process again enforces systematicity in selection a set of bridges which represent the mapping. Nodes in the source that have received activation but do not have a bridge associated with them are selected for transfer to the target. The evaluation & use stage checks the nodes linked to the selected nodes for bridges. If a bridge exists, the corresponding node in the target is linked by the same relation to the newly created target node. If a bridge does not exist, a "skolem" node is built and is then treated as if a bridge had been found. This is done for each link connected to the selected source nodes. In this example, the generalized "KISS" node is transferred over to the target situation due to activation gained by spread over the *cause* link. Since bridges exist from the *agt* and *pat* of the "KISS" node, the nodes at the other end of the bridges are linked to the new node.

The evaluation & use stage checks the transferred information to see if the current goals have been attained. In this example, since a "CAUSE" link has been transferred to the target, the goal has been satisfied. The action predicted by ASTRA in this situation is that "Romeo will kiss Juliet". If the transferred information merely elaborates the target representation and the goal is not satisfied, then the system will re-activate the new representation of the target to see if any further information can be conjectured.

5. RELATED WORK

In Sun [11], a connectionist model of rule-based reasoning is developed, called CONSYDERR. This connectionist system has two levels: one is a connectionist network with a localist representation, and one is a connectionist network with a distributed representation. The two networks are linked by connecting the distributed nodes which make up a concept with the localist version of the concept. A rule-defining "knowledge link" is simply a weighted connection between the antecedent and consequent of the rule. The representation

differs in that the knowledge links emanating from the localist nodes are duplicated in the distributed network so that every node in the distributed representation of the local node has a knowledge link to the group of nodes in the distributed representation of the consequent node. This arrangement is beneficial in that nodes which share features with a node in the antecedent of a rule will also have some ability to fire that rule, based on the similarity in their representation. However, this implementation has the “multiple instantiation” problem common in connectionist networks for high-level reasoning [12]: The rule cannot be applied to two instances of the antecedent because there is only one node representing the consequent. The problem is solved in ASTRA by having rule-firing create a new instance of the consequent node.

Falkenhainer [3] describes an approach to explanation and theory formation that attempts to unify deduction, assumption, induction, and analogy. The reasoning methods are discriminated not by mechanism, but identifying different types of similarity with the target. Deduction is defined as a complete match of identical features; Assumption is a partial match of identical features, which if augmented by a finite set of consistent assumptions will allow an explanation to be deduced; Induction is the case where matches are made between features with a close common generalization; and Analogy, where a wide range of features and relations are matched. This approach differs from ASTRA in that it does not take discriminate on the source retrieved, but on the matches that are produced during mapping. Falkenhainer’s implementation is entirely symbolic and discrete approach to integrated reasoning, and thus is polarized with the ASTRA work.

Kokinov [1,2] has developed a system called AMBR in which deduction, induction and analogy are also treated as slightly different manifestations of associative memory-based retrieval. As with the continuous analogical reasoning theory, the differences of deduction, induction and analogy lie only in the outcome of the associative retrieval process. Kokinov’s work differs from the ASTRA work in the knowledge organization and processing mechanisms. The knowledge representation used in AMBR is entirely symbolic, based on a frame system. However, connectionist networks are dynamically built to select the best interpretation or mapping for the analogy. The processing mechanisms include a relaxation search (spreading activation), marker passing, and traditional procedural code. It is not clear as to the extent any interactions between stages play a role in the processing of an analogy. Since networks are dynamically created to select a mapping, much of the correspondence information generated by source retrieval may be lost.

Psychological experiments by Kokinov[2] have demonstrated common priming effects in tests of deduction, induction and analogy. Because of these common effects, Kokinov concludes the claim that deduction, induction and analogy are performed by a single uniform retrieval mechanism. Burstein and Collins [13] also conclude that the type of knowledge retrieved determines the particular line of inferencing produced. The work presented here is further evidence of how a single mechanism can be used to produce the different reasoning behaviors.

6. CONCLUSIONS

We have presented a hybrid connectionist-symbolic model in which not only can deductive, inductive, and analogical reasoning behaviors be produced, but integrated to work together during problem solving. In the example, deductive reasoning is used to enhance the representation of the target analog. Inductive generalization is used when a number of similar source analogs are activated, resulting in a new structure that is composed of their common features. This type of induction has been shown to be useful in analogical reasoning also serves as a learning and memory organization mechanism [14].

The ASTRA system is written in CommonLisp and runs on Sun and Macintosh workstations, and Symbolics and Explorer Lisp machines. The system has been tested in simple domains such as the examples given in this paper which have only 90 nodes and 300 links, to large domains such as database of plays which require 3076 nodes and 13148 links. Future work will integrate ASTRA into a problem solving system to more fully test the theory. Another area of future work involves simplifying the model by implementing the knowledge representation and processing mechanisms in an completely connectionist framework. Realizing ASTRA in a connectionist network may further extend its natural ability to handle novel input situations. The use of induction as a memory structuring mechanism will also be investigated. We believe that the integrated reasoning features of ASTRA can produce memory organization results comparable to Pazzani's OCCAM program for integrating similarity-based, theory-driven, and explanation-based reasoning [15].

7. ACKNOWLEDGEMENTS

Discussions with Jeff Shrager and Brian Falkenhainer were helpful in the early development some of the ideas in the paper. Discussions with John Barnden and the members of the Metaphor and Analogy group at New Mexico State University were helpful in specifying the processes involved in the ideas presented.

8. REFERENCES

1. B.N. Kokinov, "Associative Memory-Based Reasoning: How to Represent and Retrieve Cases," *Artificial Intelligence III: Methodology, Systems, Applications*, 51- 58 (1988).
2. B.N. Kokinov, "Associative Memory-Based Reasoning: Some Experimental Results," Proceedings of the Twelfth Annual Meeting of the Cognitive Science Society, 1990, pp. 741-749.
3. B. Falkenhainer, "A Unified Approach to Explanation and Theory Formation," *Computational Models of Scientific Discovery and Theory Formation*, 157-196 (1990).
4. T.C. Eskridge, "A Continuous Approach to Analogical Reasoning," Proceedings of the Third Rocky Mountain Conference on Artificial Intelligence, Denver, CO, 1988.
5. T.C. Eskridge, "Principles of Continuous Analogical Reasoning," in *Journal of Theoretical and Experimental Artificial Intelligence* 1 (3), 179-194 (1989a).
6. T.C. Eskridge, "Continuous Analogical Reasoning: A Summary of Current Research," Proceedings of the DARPA Workshop on Case-Based Reasoning, Pensacola Beach, FL, May 20-23, 1989b, pp. 253-257.
7. T.C. Eskridge, "A Hybrid System for Analogical Reasoning," submitted to *Advances in Connectionist and Neural Computation Theory, Vol 2: Connectionist Approaches to Analogy, Metaphor and Case-Based Reasoning*.
8. M. Mitchell and D.R. Hofstadter, "The Emergence of Understanding in a Computer Model of Concepts and Analogy-Making," *Emergent Computation*, 322-334 (1990).
9. D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Volume 1: Foundations*, MIT Press, Cambridge, MA (1986).
10. J. Anderson, "Spreading Activation," *Tutorials in Learning and Memory*, 61-90 (1984).
11. R. Sun, "Connectionist Models of Rule-Based Reasoning," to appear in Proceedings of the 13th Cognitive Science Conference, Chicago, IL, 1991.

12. J.A. Barnden and J.B. Pollack, "Introduction: Problems for High-Level Connectionism," *Advances in Connectionist and Neural Computation Theory, Vol 1: High-Level Connectionist Models*, (1991).
13. M. Burstein and A. Collins, "Modeling a Theory of Human Plausible Reasoning," *Artificial Intelligence III: Methodology, Systems, Applications*, (1988).
14. M. Gick and K. Holyoak, "Schema induction in analogical transfer," *Cognitive Psychology* **15**(1), 1-38 (1983).
15. M. Pazzani, *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. LEA, (1990).

A FIRST ORDER THEORY FOR REPRESENTING SPACE IN KNOWLEDGE-BASED SCENE GENERATION ¹

Enrico Giunchiglia and Alessandro Armando

Mechanized Reasoning Group
DIST - University of Genoa
Via Opera Pia 11A, 16145 Genoa, Italy
enrico@dist.dist.unige.it
armando@dist.dist.unige.it

ABSTRACT

The key idea of the work described in this paper is to use a declarative representation of space (based on first order logic) as well as a procedural one (based on a semantic network). The link between the two representation is made by semantic attachment [1], defining the semantic network to be the intended model of the first order theory. The paper describes how we have modeled space, both syntactically and semantically.

1. INTRODUCTION

In the past several space representations have been proposed [2, 3, 4, 5, 6]. Early papers (for example [7, 8]) and recent papers (notably [9, 10]) have been written on this topic by some members of our group.

The aim of this paper is to give some further leading ideas and results which seem useful in order to build a suitable representation of space for systems able to reason in a “human-like” way. The key idea of the work described in this paper is to have a first order description as well as a semantic network [11] (containing a more extensive description) of the same space. The two world are related defining the semantic network to be the model of the first order theory: the first order objects and relations are linked to the objects and relations of the semantic network by means of semantic attachment [12, 1].

We are not faced with the philosophical reasons of our approach or of others; this topic has been already faced in [8, 9]. Neither are we faced with the aim of the complete work and its possible applications (see [7, 10]). The reader is recommended to read that papers to have a complete view of the historical developments and related work of our theory.

¹This work is part of a joint project among all the members of the Mechanized Reasoning Group. All of them are thanked for the invaluable support and discussions we took together and for allowing us to build on their previous work.

The paper follows this path: in the next section we present some comprehensive ideas which have driven the system implementation. In the third section we try to formalize some ideas on how it is possible to reason about space when generating a scene, starting from a human-like ambiguous description, made for instance in natural language. We propose a not complete (and not completely described) first order theory enriched with first order default rules [13]. In the fourth section we focus on a representation of space (even using multiple valued/fuzzy semantics [14, 8]) which can be seen as an extensional model of the logic/ syntactic theory. Finally in the fifth section it is shown how the “semantic attachment” between the first order theory and procedural reasoning is performed.

2. A PROCEDURAL/DECLARATIVE APPROACH TO SPACE REPRESENTATION

In the past, when speaking about data representation, people were concerned with new data structures, of whichever complexity. From this point of view, data were seen as a declarative part, modified and elaborated by programs. New trends in computer science research tend to eliminate the dichotomy data/ procedure (or declarative/ procedural information storage) in order to create new, more abstracted structures, which join the two aspects (LISP, PROLOG and present research on data abstraction are only some trivial examples).

Human knowledge is a special kind of information (even if more and more complex), whose representation (namely knowledge representation) should be faced with the same criteria used for the “classical” data representation [11, 15]. This consideration led us to face the problem of space representation from both a procedural and declarative point of view. This state of mind was enforced by the consideration that in order to build a system with human-like with an approach as general as possible, a purely algorithmic/ not structured approach would not have been sufficiently powerful. Human reasoning (about space and object positioning) is very structured. At least we can recognize a first qualitative step of general evaluation of the problem and a final quantitative step of numerical values handling.

The reasoning we propose is largely qualitative. Our approach, extensively adopted by some A.I. researchers, for instance in [16, 17, 3], is based on the following considerations: first, it allows to reach conclusions with very little information available and, consequently, hard to formalize with quantitative models; second, qualitative reasoning can be very effective to reach approximate conclusions, sufficient in everyday life, at the right level of detail.

A second consideration, which directly followed from the previous, led us to the use of logic. Logic allows a well formed definition of the problem and the possibility of moving within an extensively studied and well known environment. In this approach first order rules implement the syntactical reasoning

about space and are in some way related to the system qualitative reasoning. Procedural reasoning can be seen as the semantics of the first order theory of space and is in some way related to the system quantitative reasoning (even if it is quite difficult to define an univocally defined border between qualitative and quantitative reasoning, logic and procedural implementation). The link between the two worlds is made by a sort of “semantic attachment” [12, 1] which allows the system to switch on semantic reasoning when syntactic reasoning does not seem sufficient.

More specifically the representation of space we propose is compounded of three parts:

1. First order theory of space (written in some logic language, namely PROLOG). At this level deduction is performed both syntactically (as in any “classical” proof checker) and through semantic evaluation of functions and predicates (performing the “semantic attachment” informally stated in the previous section. This topic is deeply faced in the next section).
2. Semantic network (which implements the domain of the interpretation of the extensional model of space). That is, space is represented through a graph where the nodes are the objects being in the scene and the links are marked by the (natural language) spatial relations holding between them.
3. Procedural evaluation of the semantic network (written in some procedural language, namely the C-language). Such procedural evaluation is performed by a *deduction supervisor* which knows all the couples:

< syntactic object, intended meaning >

(the intended meaning of a syntactic object is a member of the domain D (of the intended model [18]) if the object is a constant, and a procedure, if the object is a functional or predicative symbol). Evaluating a syntactic expression corresponds to the activation of the associated (semantic) procedure given the semantic meaning of its sub-expressions. The equality between the syntactic object whose intended meaning is the computed result and the expression being evaluated, is then asserted in the first order, ground theory. Taking an example by arithmetic, attaching the numbers “2,3,5” respectively to the first order constants “two, three, five” and the procedure computing the sum of two numbers to the function symbol “plus”, evaluating “plus(two,three)” makes “plus(two,three)=five” being asserted in the theory.

3. FIRST ORDER REASONING ABOUT SPACE AND OBJECT POSITIONING

The system accepts, as input, natural language-like scene descriptions. In particular input phrases are codified in a first order language (with sorts). For instance the phrase “there is a pen on the table” is codified:

ON(a,b) where:

"ON" is predicate with two arguments

"a" is a variable of sort "pen"

"b" is a variable of sort "table"

Logical reasoning is performed in two steps:

1. First it is tried to disambiguate the input relation that is to define which among all the possible spatial meanings of the input proposition (for instance "on") is the most "suitable". In this step default reasoning plays a basic role [7].
2. Second, it is tried to deduce more information about the predicates derived in the previous step. For instance it is obvious that if RIGHT(a,b) holds also LEFT(b,a) holds. This part of the theory is completely independent of the previous and can be seen as a first order description of the relations which exist among object positions.

The first step is mostly performed syntactically (even if the holding of several predicates is tested semantically) while the second is mostly performed procedurally.

First step rules have the form:

$$\begin{array}{l} P1(x1,x2,\dots) \text{ and } P2(y1,y2,\dots) \dots\dots \\ \implies \\ P(\dots\dots) \text{ and } Q(\dots\dots) \dots \end{array}$$

where the predicates P1(...), P2(...) ... can be procedurally evaluated on the basis of two different kinds of information: i) the a-priori knowledge of the world (this problem is not faced here, for a deeper insight see [7]) and ii) the current state of the world, that is the semantic network above introduced. Note that the semantics of the first kind of predicates is intensional, while that of those of the second kind is extensional. The major difference between the predicates in our formalization and the standard logical ones lies in their semantics (that is in their procedure of evaluation): the nonlogical symbols of a standard logical language (that is functions and predicate symbols) are taken to be independent and primitive, on the contrary our predicates can be definitionally related to each other.

As far as default reasoning is concerned each rule can be generally described [19] as:

$$\{wff_1\} : M\{wff_2\} \implies \{wff_2\} \quad (1)$$

where M is to be read "it is consistent to assume". The whole rule is to be read in this way "if $\{wff_1\}$ holds and it is consistent to believe that $\{wff_2\}$ holds then infer $\{wff_2\}$ " (note the difference between the implication sign (\implies) and the assumption sign (\implies)). Some considerations must be made on the (1): $\{wff_1\}$ is evaluated testing the knowledge base and its holding is the basis for the activation of a default; $M\{wff_2\}$, that is the considerations of

consistency with the real, present world, is evaluated testing the semantic network. It tests the compatibility of what has been inferred previously with the output of the knowledge base and decides if backtracking has to be activated when exceptions arise.

These rules are an attempt to formalize (to give a cognitive model of) common sense reasoning about space. They do not take in consideration all the possible combinatorial situations. In fact some of them are cognitively impossible and are automatically impossible for how the knowledge base has been built. To understand what they mean it is necessary to apply them to real situations.

An interesting example may be:

```
RIGHT(typewriter, nick-nack)
    ===>
    H_CONTACT(typewriter, table) and
    H_CONTACT(nick-nack, shelf) and
    H_CONTACT(table, floor) and
    H_CONTACT(shelf, floor)
```

As it can be seen we first deduce that the typewriter is horizontally supported by the table and then we recursively deduce the positions of all the inferred objects till the border of the environment [7, 9]. The rule applied when deriving H_CONTACT(typewriter, table) is a simplified form of the “rule of the independent typical positions” [7]. Formally, it can be so described:

```
W_CONC(s,o)

/* the input conceptualization is a weak */
/* conceptualization (derived from on the */
/* right, in front of, near, ...).      */

    and
TP(o)

/* the object has a typical position */
/* in the defined environment        */
/* (evaluated intensionally on the   */
/* a_priori data base)                */

    and
NEAR_POSIZ(s,o)

/* the subject and the object of the */
/* conceptualization are near to each */
/* other if positioned in their own  */
/* typical positions                   */
```

```

/* (evaluated extensionally on a      */
/* possible state of the semantic     */
/* network)                            */

    and
(((TP_H(s) and TP_H(o)) or
    (TP_V(s) and TP_V(o)))

/* if positioned in their own typical position */
/* both the subject and the object are supported */
/* either by an horizontal or vertical surface */
/* (evaluated as TP(o))                      */

    and
not COMMON_MATRIX(s,o,m)

/* an object m which is the common matrix for the */
/* object and the subject does not exist. The common */
/* matrix of n objects is an object to which all the */
/* objects refer when positioned in their typical */
/* position                                          */
/* (evaluated as TP(o))                            */

    ==>
    TP_CONC(s,s,kb_tp_obj(s)) and
    TP_CONC(o,o,kb_tp_obj(o))

/* both the subject and the object are positioned in */
/* their typical position, extracted from the data */
/* base a_priori with the function kb_tp_obj(o) */

```

Note that we have supposed that the nearness of the two typical positions holds. Of course this is not always true, the contrary may happen even in this case, depending on how the table and the shelf are positioned. In this case the analysis changes: we position the object in its position but we do not know anything about the subject position. Thus we have:

```

    W_CONC(s,o) and TP(o)
    and
(((TP_H(s) and TP_H(o)) or
    (TP_V(s) and TP_V(o)))

    and
    not NEAR_POSIZ(s,o)
    and
not COMMON_MATRIX(s,o,m)

```

```

====>
TP_CONC(o,o,kb_tp_obj(o))
  and IND_POS(s)

/* the object is positioned in */
/* its typical position, for   */
/* what concerns the subject,  */
/* we are not able to decide   */
/* and ask                      */

```

This last rule is a subset of the “rule of the unknown subject position”.
As far as the second step is concerned examples of rules are:

```

forall x. forall y. (RIGHT(x,y) --> (LEFT(y,x))
forall x. forall y. forall z.
    (H_CONTACT(x,y) --> not H_CONTACT(x,z))

```

Two considerations must be made:

1. This theory is largely incomplete. This is due to the extreme complexity of the problem which makes the problem unsolvable with this approach. To point out this fact it is sufficient to think to the infinite mutual positions that two objects may have.
2. As a consequence of the above consideration, in this step most reasoning is performed procedurally, on the basis of the semantic network; only simple cases, such as those stated above are treated symbolically. Thus, this topic is treated in the following sections.

4. SPACE AS A SEMANTIC NETWORK

A first important consideration must be made. We have approached the problem of space representation and object positioning from two points of view (with double valued and multiple valued logics). As far as syntactically reasoning is concerned, till now only the double valued version has been implemented; as far as the the semantic network and procedural reasoning are concerned both versions exist. In the following we will treat space as a linguistic variable [14], that is we describe the multiple valued approach. This is not correct (a double valued theory must have a double valued interpretation). We describe space in this way because we think it is more interesting; in the following, when we refer to the semantic network as the interpretation of the previous described theory, assume that we refer to the multiple valued version.

Let us focus on how space is represented. In the following we give only some ideas (for more details see [9, 10]).

When working about space representation we noted that, in an everyday discussion, object positions are nearly always defined relatively to the positions of other objects whose positions are recursively ill-defined. On the other hand, people seem able to infer the positions of objects with respect to the environment reference system. Thus, for instance, people are able to say that, if “pen1 is on the right of book2” and “pen3 is on the left of book2” then it is impossible that “pen1 is on the left of pen3”. Moreover, when trying to position objects in a limited space people are also able to shift them maintaining all the known (ambiguous) constraints. Our solution has been to memorize the single compatibility functions within a graph whose links represent object-subject couples.

SUBJ -----> OBJ

Working in this way we are able to build all the absolute references, walking through the graph and composing functions using a generalized AND operator, but we memorize space within a structure which is strictly related to its natural language description [10]. Every link is marked by an input (ambiguous) constraint and has associated a compatibility function which describes the set of all the possible mutual object-subject positions with their compatibility/ possibility/ truth values. Again, how compatibility functions are built is largely explained in [10]; here only some notes are given.

First of all, in the fuzzy approach we assume that all the input phrases are elliptical; that is that some key words, necessary to understand the meaning of the input phrase, due to the common use/position of the objects, are left unmentioned. So, for instance, “book on the table”, means not only that the book is horizontally supported by the table but also that, maybe, is in its centre (or in a side ...) with an orientation which guarantees a high degree of equilibrium ... Of course the words we suppose are left unmentioned are strictly dependent on the objects mentioned (i.e. a book is usually in the centre of the table while the phone is maybe on one side, more usually the right side). The functions which describe the mutual positions of the object and the subject of the spatial prepositions have some interesting features:

- i) they depend on the given spatial preposition, in fact any spatial preposition refers to different object parts or characteristics (for instance **on** relates the volume of the subject to a surface of the object while **in** relates the two volumes);
- ii) the influence of an object on the function shape depends on its syntactic role in the spatial relationship (whether it is subject or object of the spatial relation);
- iii) each function is built independently of any contextual check, the “contextual” space state is taken into account only when the overall “free space” compatibility function is synthesized and iv) the relation (defined

a_priori) between the above parameters and the function values, because of the very nature of common sense, is necessarily fuzzy and not crisp.

5. PROCEDURAL REASONING ABOUT SPACE

Procedural reasoning is performed through a set procedures which are activated by the “deduction supervisor” and modify or read the state of the semantic network. These procedures can be divided in two classes:

1. The first are activated when a new predicate (such as `H_CONTACT(typewriter, table)` is deduced) and modify the semantic network. These procedures are responsible of maintaining a full compatibility between the theory and the semantic network.
2. The second are activated when the validity of a predicate must be tested. They test if the predicate is compatible with the semantic network.

A complete description of their behavior is beyond the aim of this work; see [10]. An example can be the analysis given in section 3. In that example the same input may give two different answers depending on the semantic evaluation of the predicate `NEAR_POSIZ(s,o)`. This may be a good example of how semantics and syntactics cooperate and how the semantic attachment is performed. In fact the deduction supervisor, when working on `NEAR_POSIZ`, understands that the evaluation must be made procedurally and switches to the semantic level.

A very interesting note. Most backtracking is activated at this level. In fact, when syntactically reasoning, the system activates a lot of default rules. This may be misleading and give some wrong deductions. The absurd is mostly verified at this level, usually when it is tried to deduce a new predicate which is inconsistent with the network state. Note that the backtracking routines are also able to give some information about which default has been erroneously activated.

Finally, it is worth noting that our representation of space completely fits with standard extensional semantics definition (as, for instance in [18]). Let us consider the double valued version of our semantic network:

- `D`, that is the domain of the interpretation, that is space described, for instance by means of cartesian coordinates is

$$D = R \times R \times R$$

- any constant, that is any object instantiated, is associated an element belonging to `D`, that is a triple (x,y,z) ;
- any variable, that is any object not instantiated is assumed to vary within `D` (we do not know where it is);

- any function is assumed to get values in $D \times D \dots$ and to take values in D ;
- the standard logical operators are assumed to have their propositional value. So for, instance, the holding of a set of input phrases is interpreted as the set intersection of all the single sets;
- the predicates we have defined in our theory are interpreted as subsets of D . For instance, $H_CONTACT(\text{pen}, \text{table})$ is associated the set of all the points which are above the upper surface of the table.

6. CONCLUSIONS

We think that ten pages are too few in order to give a complete explanation of how we have modeled space and what we think about logic and knowledge representation. We have tried to give only some general ideas, never explaining the details of the formalization. This has probably resulted in a sometimes not clear, always too brief and not precise, explanation. We hope that people reading this paper do not care of style and formal matters and understand the underlying ideas. A paper with a more precise and complete formalization is forthcoming.

REFERENCES

- [1] P. Pecchiari. Meccanizzazione del concetto di modello di un dimostratore interattivo. Master's thesis, University of Trento, Dept. of Mathematics, 1990. IRST-Thesis 9009-15.
- [2] K. Lynch. *The image of the city*. MIT Press, 1960.
- [3] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129-153, 1978.
- [4] D. McDermott and E. Davis. Planning routes through uncertain territory. *Artificial Intelligence*, 22:107-156, 1984.
- [5] B. Faltings. Qualitative kinematics in mechanisms. In *Proc. IJCAI conference*, pages 436-442. International Joint Conference on Artificial Intelligence, 1987.
- [6] E. Davis. A framework for qualitative reasoning about solid objects. In *Proc. of the workshop on space telerobotics*, pages 369-375, 1987.
- [7] A. Adorni, M. DiManzo, and F. Giunchiglia. From descriptions to images: what reasoning in between? In *Proc. 6th European Conference on Artificial Intelligence*, September 1984.

- [8] M. DiManzo and F. Giunchiglia. A representation of space for knowledge based scene generation. In *Cognitiva 85*, June 1985.
- [9] F. Giunchiglia, C. Ferrari, P. Traverso, and E. Trucco. Understanding scene descriptions by integrating different sources of knowledge. Technical report, Dep. Comp. Science, University of Genoa, October 1990. Technical Report DIST-90-54. To be published in the International Journal of Man Machines Studies.
- [10] A. Armando, A. Cimatti, E. Giunchiglia, and P. Traverso. A system for natural language driven 3-d generation. Technical report, Dep. Comp. Science, University of Genoa, January 1991. Technical Report DIST-91-03.
- [11] R.J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer IEEE*, October 1983.
- [12] R.W. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13, 1980.
- [13] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13, 1980.
- [14] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning i-ii-iii. *Information Sciences*, 8-9:199-249, 301-357, 43-80, 1975.
- [15] W.A. Woods. What's important about knowledge representation? *Computer IEEE*, October 1983.
- [16] K.D. Forbus. Qualitative process theory. *Artificial Intelligence.*, 24:85-168, 1984.
- [17] J. Dekleer. Qualitative and quantitative reasoning in classical mechanics. In P.J. Winston and R.H. Brown, editors, *Artificial Intelligence: an MIT perspective*. MIT Press, 1979.
- [18] Z. Manna. *Mathematical theory of computation*. Mc Graw hill Book Company, 1974.
- [19] D. McDermott and J. Doyle. Non-monotonic logic ii. *Artificial Intelligence*, 13(1), April 1980.

LEARNING INTERMEDIATE CONCEPTS IN CAUSAL TREES: A DIRECT METHOD

Jean-Louis Golmard

Inserm U 194 et Département de Biomathématiques,
91, Bd de l'hôpital, 75634 Paris, France.
email: Golmard@frsim51

ABSTRACT

We propose an empirical but a direct method used to learn partially unknown causal tree structures. More specifically, we suppose that the intermediate layer of a three-layered causal tree is completely unknown. A set of examples where the root node and the bottom layer nodes are observed is available. The principle of the method is to compute a correlation matrix which is block diagonal according to the model. This structure is then found using a clustering algorithm. A simulation experiment highlights the potential usefulness of the method in practical situations. Finally, open problems are briefly reviewed.

1. INTRODUCTION

This paper describes a method used to solve the following problem: one supposes that a domain universe can be represented as a causal tree with three layers, the structure of which is partly unknown. More precisely, the root node of the tree and the bottom layer correspond to observable random variables, while the intermediate layer is unknown, that is, nor the number of intermediate nodes nor their linked nodes are known. The problem is then to find the hidden structure of the causal tree from a set of samples which consists in observed values of the root node and the bottom layer nodes. In a previous work [Golmard and Mallet 1989], a method for estimating the probabilities of a three-layered causal tree with a hidden intermediate layer has been described. The structure of the tree was supposed known. This current paper therefore completes the preceding one, although the method described here is to be used before the one of the previous work.

The interest of learning hidden tree-structured causal structures has already been outlined [Pearl 1985, 1988]. The advantage of using tree structures comes partly from the simplicity of the required computations. Furthermore, they allow simple interpretations of the causality relationships. Intermediate concepts are usually used in many domains, like the medical domain, where they are often called "syndromes", because they facilitate the description and the memorization of cases, and thus, the reasoning process of human brains. As already stated in our previous work, syndromes may be viewed as ill-defined concepts, since their "structure", that is the signs entering their definition, is known, but it is difficult to assess their presence, given a case, when a part of these signs is observed. Learning the structure of the causal tree can be interesting in several types of situations: the domain may be almost unknown, so the problem is to learn the structure, or the expert may have some doubts about a part of the structure he can detail. In this latter case, the learning method may be aimed at validating an already stated causal tree model. The theory of the comparison between a learned structure and a theoretical one is not well established, and therefore research work remains to be done in this field. Another possible use of a structural learning method is to find a good approximation of a more complicated underlying model.

Methods for learning causal tree structures have already been described. The powerful algorithm of [Chow and Liu 1968] is to be recommended when all the variables are observable. Spiegelhalter and Lauritzen [1989] have proposed a method based on sequential Bayesian estimation for general probabilistic networks. Roizen and Pearl [1986] have described an iterative algorithm for estimating the probabilities when a part of the variables are unobservable and the causal tree structure is known. Their method is related to the stochastic approximation algorithms (see, for example, [Kushner and Clark 1978]). The algorithm we proposed in the same context [Golmard and Mallet 1991] is based on maximum likelihood estimation. When the structure is not known, the problem is then to learn “hidden causes”, or to learn structures. An algorithm for learning hidden causes in causal trees has been proposed by Pearl [1985, 1988]. However, this algorithm involves many comparisons between triples of variables, and is not robust relatively to the randomness of experimental data, as noticed by Pearl himself. Finally, our method is related to the clustering methods, which cannot be all cited here.

The method will be described in section 2. In order to evaluate the accuracy of the method in various experimental setting, we have performed a simulation experiment. The results of this experiment are detailed in section 3. Finally, possible directions of future works are outlined in the conclusion.

2. METHOD

2.1 NOTATIONS AND PROBLEM FORMULATION

The notations we propose emphasize the three layered structure of the causal tree, and they are related with the diagnostic problem field. The data consist of a set of samples. For each sample, the values of the diagnosis and the values of the set of signs are provided. The vector of the signs is noted S . We note D the random variable “diagnosis”, and d one of its possible values, and similarly for all the random variables. We suppose that the intermediate (unknown) layer of the causal tree is composed of k hypotheses, noted $H = (H_1, \dots, H_k)$. The value of k is unknown. Each hypothesis H_i is the direct cause of n_i signs, denoted $S_{i,n1}, \dots, S_{i,ni}$. The structure being a tree, each sign is linked with exactly one parent hypothesis. The signs may then be partitioned into several groups, each group corresponding to one hypothesis. The problem of finding the “true” hidden causal tree structure from the data may then be formulated as the problem of finding the “true” partition of the signs. The proposed method, however, is not guaranteed to find the true structure: the probability of finding the true structure will be dependent of many features, three of which have been used in the simulation experiment described in section 3.

2.2 PRINCIPLE

We suppose that all the random variables are of binary type. The probability distribution of a causal tree is then expressed as:

$$P(d, h, s) = P(d) \prod_{i=1}^k [P(h_i) \prod_{j=1}^{n_i} P(s_{i,j} | h_i)] \quad (1)$$

From expression (1), it is easy to compute the conditional probabilities of the signs when the diagnosis is known. We have:

$$P(s | d) = \prod_{i=1}^k P(s_{i,1}, \dots, s_{i,n_i} | d) \quad (2)$$

Expression (2) illustrates the conditional independence of the groups of signs, since the global probability function is divided into a product of k probabilities. Two signs which do not belong to the same group (they have not a common cause), are independent conditionally to the diagnosis d . This independence property is the key of the method. In fact, computing the conditional covariance of two signs $S_{i,j}$ and $S_{t,u}$, we get:

$$\begin{aligned} \text{cov}(S_{i,j}; S_{t,u} | d) &= 0 \text{ if } i = t \\ &\neq 0 \text{ if } i \neq t. \end{aligned}$$

In matrix notation, $\text{cov}(S | d)$ is the matrix of the conditional covariances of couple of signs. The proposed method could be used from each such conditional covariance matrix. However, in order to obtain an unique covariance matrix, a weighted sum of the conditional covariance is computed as follows:

$$\text{cov} = \sum_{d=1}^2 \text{cov}(S | d) P(d)$$

Note that the conditional covariances are not supposed to be equal, and therefore the final covariance is not an estimate of any individual covariance. Its interest is just to resume in one matrix the two conditional covariance matrices. Finally we compute the correlation matrix, using:

$$\text{cor}(i, j) = \frac{\text{cov}(i, j)}{\sqrt{\text{cov}(i, i) \text{cov}(j, j)}}$$

This matrix is block diagonal. This structure may be found using a principal component analysis on the variables (see, for example, [Mardia et al. 1979] or [Morrison 1976]), since the k largest eigenvalues must correspond to the k blocks of the correlation matrix. The algorithm we actually use is described below.

2.3 IMPLEMENTATION

2.3.1 Correlation matrix computation

Each sample consists in one value of the diagnosis node and N sign values. We suppose N_{IND} samples are available and we first compute the usual experimental covariance matrix for each class of diagnosis. Since we are dealing with binary variables, there are two classes, each class corresponding to an experimental conditional covariance matrix. Let us suppose that N_d samples are in the class $\{D = d\}$, $d = 1, 2$. The final covariance matrix is computed as follows:

$$\widehat{\text{cov}}(S) = \frac{\sum_{d=1}^2 (N_d - 1) \widehat{\text{cov}}(S | d)}{N_1 + N_2 - 2}$$

2.3.2 Clustering method

The experimental correlation matrix deduced from the above covariance matrix is the input of a clustering program. We have actually used the VARCLUS procedure of the SAS statistical package. The clustering algorithm can be informally described as follows:

1. Initialization: (1 cluster of N variables)

- a) Nbcluster := 1
- b) Compute the eigenvalues of the initial correlation matrix
- c) Sort the eigenvalues by decreasing order. Let λ_1 be the second eigenvalue.
- d) Denote $\lambda_{\max} = \lambda_1$, $c_{\max} = 1$

2. loop: repeat while $\lambda_{\max} > 1$ and Nbcluster < N

a) Split the cluster c_{\max} into two clusters, corresponding to the two greatest eigenvalues, according to the correlation of the variables with the two principal components, respectively. Rearrange all the variables (not necessarily contained in the cluster c_{\max}) by assigning them to the cluster the principal component of which the correlation is the highest.

b) Nbcluster := Nbcluster + 1

c) Compute the eigenvalues of the modified or created clusters and sort them for each cluster, let λ_i be the second eigenvalue of the i th cluster, $i \in \{1, \dots, \text{Nbcluster}\}$.

d) Let $\lambda_{\max} = \sup(\lambda_1, \dots, \lambda_{\text{Nbcluster}})$, and c_{\max} the corresponding cluster number.

e) end of the loop.

A summary of the results listing based on an example is shown in figure 1. The 16 initial variables are finally divided into 4 clusters. Note that, in the step 4, variable 19 has been removed from cluster 3, and is eventually a member of the last created cluster 4. The correlations between variables and clusters are not displayed, because of the lack of space, but the correlation variable 14 - cluster 3 is 0.1688 at step 3, and 0.0560 at step 4, while the correlation variable 14 - cluster 4 is 0.4745.

Note that the method we propose does not need to specify the number of clusters to be found. The result of the procedure is a partition of the initial set of N variables into 1 to N clusters. We are only interested here in the structural part of the learning. The estimation of the probabilities could be done using the method described in [Golmard and Mallet 1991].

Fig. 1. Summary of a result listing.

Nbcluster	Cluster	Members	Variables	Second eigenvalue	Cluster to split
Step 1					
1	1	16	1-16	2.2891	1
Step 2					
2	1	11	5-13, 15-16	1.8187	1
	2	5	1-4, 14	0.9494	
Step 3					
3	1	7	5-8, 13, 15-16	1.6770	1
	2	5	1-4, 14	0.9494	
	3	4	9-12	0.8490	
Step 4					
4	1	4	5-8	0.7450	-
	2	4	1-4	0.7292	
	3	4	9-12	0.8490	
	4	4	13-16	0.7080	

Notation: i-j means $\{k \mid i \leq k \leq j\}$.

3. A SIMULATION EXPERIMENT

The learning method described in the previous section is not guaranteed to find the true hidden structure in all the encountered situations. The asymptotic convergence of the method seems very probable, since the experimental conditional covariance matrices converge toward the "true" ones, but the fact remains to be proved. Furthermore, simulations are useful for illustrating the kind of results which can be found, depending of various experimental features.

3.1 SIMULATION SETTING

We have tried to minimize the number of parameters required to describe the simulation experiment. All the variables are binary ones.

. The structure of the causal tree is then resumed by only one parameter, which is denoted NS (for Number of Signs by hypothesis). We suppose that the causal tree is composed by a root node D, NS hidden intermediate hypotheses H_1, \dots, H_{NS} , and that each hypothesis H_i is linked with NS signs $S_{i,1}, \dots, S_{i,NS}$. Using the notations of section 2.1., we have: $k = n_1 = \dots = n_k = NS$. The final number of nodes in the simulated structures is then $1+NS+NS^2$. We have performed the experiment with two values of NS, namely 2 and 4. The corresponding total numbers of nodes are then 7 and 21, respectively.

. The quantification of the causal tree also required one parameter, denoted PSV (for Probability of the Same Value). For all the causal trees, we have:

- $P\{D = 1\} = P\{D = 2\} = 0.5$
- $P\{X_i = x / X_{C(i)} = x\} = PSV, x=1, 2$

where X_i is any node of the causal tree, except the root node, and $X_{C(i)}$ is its parent node (its cause). Note that PSV is not the probability that the values of X_i and $X_{C(i)}$ are equal, but actually PSV is the probability that the value of X_i is x , conditionally to the fact that x is the value of $X_{C(i)}$. PSV may be viewed as a measure of the strength of the links between the nodes. For each value of NS, PSV was instantiated with two values: 0.6 (weak links) and 0.9 (strong links). Thus, four probabilistic models were used in the experiment.

. Once the values of the parameters NS and PSV are provided, the probabilistic model is completely specified. The last parameter entering the simulation setting is NIND, the number of samples (Number of INDividuals) which will be used to learn the hidden causal tree structure. The simulation using has been performed using three values of NIND: 100, 1000, and 10000.

- For each value of the triple (NS, PSV, NIND), 100 sets of samples were generated according to the probabilistic model, using a random number generator of a Vax computer. For each sample, the value of the root node d and the values of the sign nodes were stored, while the values of the intermediate hypotheses were discarded.

3.2 CRITERION OF EVALUATION

For each set of NIND samples, the method described in section 2.3. was performed. The result of the method is a partition of the NS^2 signs into m non-overlapping clusters, where m is not a priori fixed, and thus varies from a set to another. The chosen criterion to evaluate the method is simply the proportion of successes of the method. We consider a result as a success when the partition resulting from the clustering algorithm is exactly the true one. In all other cases, the result is a failure. Since 100 sets were generated for each value of (NS, PSV, NIND), the number of successes for each situation is also the experimental percentage of successes of the method.

3.3 RESULTS

The results of the simulation experiment are shown in table 1. One must note that these results are experimental proportions, and thus they are random variables: we would find different results if we perform the same experiment again. The results are so contrasted (from 0 to 100) that the confidence intervals are not required for interpreting the results.

The marginal role of each parameter is clearly recovered, as expected, but the interesting part of the analysis of the results is the comparison between the influences of the three parameters.

The most important feature, at least in the ranges we have chosen, is the importance of the strength of the links, which is measured using the parameter PSV. We did not expect this result before performing the experiment. The number of samples, NIND, is known as a very important feature in any statistical learning method, and the complexity of the structure, measured by the parameter NS, was also expected to play a very important role: the number of possible partitions with 16 signs is very large, and the percentage of successes obtained when $PSV = 0.9$ and $NS = 4$ appears to us as quite satisfying.

The practical consequence of these results could be formulated as follows: it is very difficult to learn a hidden intermediate concept when the probabilistic relationships between the signs entering its definition are weak, even if the sample size is large (35% of successes with $NIND = 10000$, $NS = 4$, and $PSV = 0.6$). On the other hand, if these probabilistic relationships are strong, it is possible to learn hidden structures, even with moderately large sample sizes (100% of successes with $NIND = 1000$, $NS = 4$, and $PSV = 0.9$). An other way to state a practical advice could be: "for learning hidden concepts, choose a small number of very specific signs".

4. CONCLUSIONS

The results displayed in the previous section show that the method described may be quite efficient in practical situations. As already mentioned, the first practical interest of our method is to learn "true" causal tree structures, as far as true mathematical models are able to exist. An important tool in this context would be a method for estimating the probability that the found structure is the good one. A related problem is to study the results of our method (or any concurrent method) when the true hidden structure is not a causal tree, but a more complicated graphical model. We may be interested by this situation according to related, but different, points of view. If we are interested in proving causal relationships, we need tests between the causal tree models and the more complicated ones. If we are building diagnostic advisor systems (or expert systems), we need to know if the causal tree model is a good approximation of the more complicated model. The problem is then to estimate the accuracy of an approximation.

REFERENCES

- Chow, C.K., and Liu, C.N. (1968) Approximating discrete probability distributions with dependence trees, *IEEE Trans. on Info. Theory*, 14, 462-467.
- Golmard, J.L., and Mallet, A. (1991) Learning probabilities in causal trees from incomplete databases. *Revue d'Intelligence Artificielle*, 5, 93-106.
- Kushner, H.J., and Clark, D.S. (1978) *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, New-York.

- Mardia, K.V., Kent, J.T., and Bibby, J.M. (1979) Multivariate analysis. Academic Press, London.
- Morrison, D.F. (1976) Multivariate statistical methods. McGraw-Hill, Ney-York.
- Pearl, J. (1985) Learning hidden causes from empirical data. In: Proc. IJCAI 85, p 567-572.
- Pearl, J. (1988) Probabilistic reasoning in intelligent systems. Morgan Kaufmann, San Mateo.
- Roizen, I., and Pearl, J. (1986) Learning link-probabilities in causal trees. In: Proc. 2nd Workshop on Uncertainty, AAAI, 211-214.
- Spiegelhalter, D.J., and Lauritzen, S.L. (1989) Sequential updating of conditional probabilities on directed graphical structures, Technical report R 89-10, Aalborg University, Aalborg.

Table 1. Results of the simulation experiment

		NIND		
	100	1000	10000	
PSV = 0.6	NS = 2	19	48	100
	NS = 4	0	0	35
PSV = 0.9	NS = 2	99	100	100
	NS = 4	78	100	100

For each value of (PSV, NS, NIND), the experimental percentage of successes is displayed. The meaning of the parameters PSV, NS, and NIND is explained in section 3.1.

TRANSFORMATION-ORDERING ITERATIVE-DEEPENING-A*

Lawrence O. Hall, Diane J. Cook, and Willard Thomas
Department of Computer Science and Engineering
University of South Florida
Tampa, Fl. 33620
e-mail:hall@sol.usf.edu

Abstract

Iterative-Deepening-A* (IDA*) is an optimal search technique which requires no intermediate state storage. It is suitable for *large* search spaces. It uses a fixed order for the children of a node to be expanded. Transformation-Ordering Iterative-Deepening-A* (TOIDA*) attempts to improve upon the performance of IDA* by dynamically improving the order nodes are expanded, based on results from previous depth limits. Using the Fifteen puzzle as an example, it is shown that TOIDA* may choose the optimal fixed ordering and generally chooses a good ordering saving a substantial number of node expansions. Empirical results show that the sequential version can provide significant improvements in the speed with which a solution is discovered and no penalty in storage requirements.

1. INTRODUCTION

Search permeates all aspects of artificial intelligence including problem solving, planning, learning, decision making, and natural language understanding [7]. Because of the large state spaces that have to be searched, the performance of the search algorithm is critical to the overall performance of the artificial intelligence applications. The programming community is continually trying to improve the performance of the search algorithms and to develop new more efficient search algorithms. The various means that have been used to improve search performance include domain-specific heuristic knowledge, subgoals, and abstractions [3].

The heuristic Iterative-Deepening-A* (IDA*) search algorithm has been accepted as being asymptotically optimal in time and space over the class of best-first tree searches that find optimal solutions [3]. An optimal solution will be defined as a minimum cost solution.

1.1 ITERATIVE-DEEPENING-A*

The IDA* search algorithm consists of a series of depth first searches. On each search iteration a depth limit is set, and when a node's total cost exceeds the limit, the search of that node's subtree is abandoned. The total cost of a node n is calculated as the sum of the accumulated cost in reaching the node from the initial node ($g(n)$), and the estimated cost of continuing until a goal node is reached ($h(n)$). As with A*, in order for optimality to be assured the heuristic cost estimating function must be an underestimate of the true cost to reach the goal. On the initial search the limit is set to the estimated cost of the initial node. If the depth first search runs to completion without finding a goal node,

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 1: The fifteen puzzle

another depth first search is performed with the depth limit set to the smallest total cost that exceeded the limit of the previous search. This process is continued until a goal node is reached. For a more detailed description of the IDA* algorithm, refer to [2].

Even though IDA* provides an optimal solution in terms of cost, it can suffer a significant penalty, or realize a significant benefit, depending upon the (fixed) state transformation evaluation order used. We have developed a version of IDA* that tries to find the optimal state transformation evaluation order while finding the minimum cost solution.

2. TRANSFORMATION-ORDERING ITERATIVE-DEEPENING-A*

Transformation-Ordering Iterative-Deepening-A* (TOIDA*) is based upon the idea that by choosing a “good” fixed ordering for node expansions (or state transformations) on the final iteration substantial savings in node expansions may be achieved. Imagine that solutions may be anywhere in a final layer of node expansions and we expand children from left to right in a depth first search. If the solution is in the leftmost expansion subtree, we will not have to expand any of the nodes to the right. However, if the solution is in the rightmost subtree all of the nodes must be expanded on the last iteration possibly leading to much more work depending upon the branching factor of the problem. The idea behind TOIDA* is to choose a fixed node expansion order that brings the solution over to the left side of the search tree (allowing this simplification).

For example in the Fifteen puzzle shown in Figure 1, there are 4 possible moves from any node (or state of the puzzle). They are up, down, right or left. The blank may be moved in one of these four ways (of course in some places there will be illegal moves, because you can’t move up if you are at the top of the puzzle, etc.). In IDA* a fixed ordering is always tried from any state or node in the search space. It could be (**down, right, left, up**), which has been used by Korf in several studies [2, 3]. Again using the Fifteen puzzle as an example, there are 24 possible fixed orderings for transformations from one puzzle state (node in the search space) to the next. One of these will find the optimal solution with the fewest node expansions and one with the most possible node expansions. It is most desirable to minimize the number of expansions.

TOIDA* attempts to do this by remembering the order of state transformations that leads to the minimum h (heuristic estimate of the distance to the goal) on the next to last iteration. It then uses this fixed order on the last iteration. This order of evaluation may or may not be the same as the one it begins with.

Specifically, when the depth first search abandons a search subtree because the depth limit has been reached, the transformation ordering algorithm checks the estimated solution cost of the abandoned node. When a new minimum estimated solution cost is found,

the new *fixed* order of evaluation for the next depth-first search is set to quickly find the corresponding node. This order is determined by scanning the stack of state transformations made from the initial node, and setting the order to the relative order of the first occurrence of each state transformation. If any of the state transformations are absent from the stack, they are placed at the end of the state transformation order, keeping their same relative order as used in the search with the current depth limit.

2.1 OTHER NODE ORDERING SCHEMES

Recently node ordering has been used to increase the speed of IDA* and similar search techniques [1, 6, 5]. Powley and Korf [6] have shown that by saving an early frontier set of nodes (of the Fifteen puzzle search space), and ordering the nodes in the frontier set for expansion on the next depth-limit search, significant savings in terms of node expansions can be achieved. A frontier set of nodes in the search space is the set of nodes which have a cost over the limit on an abandoned search path. They order the nodes based upon the minimum h (heuristic estimate to goal) values of the paths that emanate from these nodes. When combined with parallel window search, they report some impressive speed increases (and decreases in the number of node expansions).

Powley and Korf originally tried saving the entire frontier set of nodes and ordering them based on the minimum h value for expansion. Although this decreased the number of node expansions, it did not decrease the actual search time. Chakrabarti et.al. [1] also report improved results, in the number of nodes expanded, with a form of depth-first search using a node ordering technique. So the benefits of ordering the nodes seems clear. TOIDA* does not perform complete node ordering since it only changes the fixed order of evaluation. However, fixed evaluation ordering achieves some of the same effects as total node ordering.

3. FIFTEEN PUZZLE

The Fifteen Puzzle consists of a four by four frame which holds fifteen movable square tiles with one blank spot, as illustrated in Figure 1. The tiles which are horizontally or vertically adjacent to the empty spot may be slid into the blank spot. The object of the puzzle is to find a sequence of tile movements that will transform the initial tile formation into a specified goal formation. Figure 1 shows the goal formation used in our experiments, which is consistent with that used in experiments by other researchers [1, 2, 4]. The optimal heuristic for the Fifteen Puzzle is the Manhattan Distance Function. The Manhattan Distance Function sums the number of horizontal and vertical grid positions that each tile is from its goal position. The value of the heuristic is this sum of the tile distances.

3.1 IMPLEMENTATION

The IDA*, TOIDA*, and Fifteen Puzzle algorithms are implemented in C on a Hypercube. Each search algorithm is implemented using exactly the same application interface so the search techniques are interchangeable. The Fifteen Puzzle algorithm is totally isolated from the search algorithms, so the searches are reusable without modification. The

Korf Puzzle #	Solution Cost	Nodes Expanded		Time (min:sec)	
		IDA*	TOIDA*	IDA*	TOIDA*
79	42	540860	411617	7:50	5:58
12	45	546344	166664	7:55	2:25
42	42	877823	225515	12:43	3:16
55	41	927212	397057	13:26	5:45
97	44	1002927	950913	14:32	13:47
19	46	1280495	1373048	18:33	19:55
94	53	1337340	136943	19:22	1:59
47	47	1411294	1523089	20:27	22:06

Table 1: IDA* vs. TOIDA*

Fifteen Puzzle was also generalized to handle any member of the square-sliding tile puzzle family. Hence, we have obtained flexibility and generality at the cost of some efficiency.

During the implementation of the search algorithms, a decision was made not to store the incremental state spaces during the search. Instead, a single state space, representing the current problem state, and a stack of state transformations applied are maintained. This decision requires that every state transformation be reversible, at least in the programming model used. As a benefit, larger problems with larger state spaces can be searched.

4. RESULTS FROM SEQUENTIAL COMPARISONS

A number of reasonable size puzzle instances have been run through TOIDA* and IDA* in our implementation. Clearly, the number of nodes expanded in our (general, but inefficient) implementation of IDA* and others will be the same, but we are interested in the time penalty of determining the search order for the next iteration in TOIDA*. As will be seen, the overhead in TOIDA* is minimal.

For an overhead comparison, the IDA* and TOIDA* search algorithms were tested against eight (of the 100) problem instances given by Korf in [2] for which the number of nodes expanded was less than 1.5 million. This limitation was necessary to get a reasonable problem set, and still have a representative sample of problems. On six of the eight puzzle instances, the TOIDA* algorithm performed better than the IDA* algorithm. The TOIDA* algorithm performance, measured by nodes expanded, ranged from 90% less to 8% more than the IDA* algorithm, with an average of 53% less. The TOIDA* algorithm showed almost identical performance increases when the index was processing time. The results of the test are shown in Table 1.

These results only indicate that the TOIDA* algorithm performs better (for these problems) than the IDA* algorithm with the state transformation evaluation order used by Korf in [2]. Since the performance of IDA* algorithm is dependent upon the state transformation evaluation order selected, the IDA* test program was modified to solve all

Korf Puzzle #	Solution Cost	Nodes Expanded		Time (min:sec)	
		TOIDA*	IDA* ¹	TOIDA*	IDA* ¹
79	42	411617	411617	5:58	5:57
			681416		9:51
			951214		13:46
12	45	166664	139857	2:25	2:01
			386895		5:35
			633933		9:10
42	42	225515	225515	3:16	3:16
			627305		9:05
			1029096		14:54
55	41	397057	397057	5:45	5:46
			663508		9:38
			931320		13:32
97	44	950913	900587	13:47	13:02
			2732664		39:34
			4570051		66:10
19	46	1373048	1057599	19:55	15:18
			2150176		31:08
			3528764		51:06
94	53	136943	136943	1:59	1:59
			757844		10:58
			1375560		19:55
47	47	1523089	1115298	22:06	16:09
			2622848		38:40
			4051749		58:40

¹ The three values displayed for the IDA* performance represent the minimum, average, and maximum values for all 24 state transformation evaluation orders.

Table 2: TOIDA* vs. IDA* (all orders)

eight problems with each of the 24 combinations of the four state transformations, and the results compared with the TOIDA* algorithm results. On four of the eight puzzle instances, the TOIDA* algorithm correctly identified the optimal state transformation evaluation order, and suffered no execution time penalty for the extra work of transformation ordering (The execution time cost for the state transformation evaluation ordering routine was below the timing function resolution of 1 second). On the remaining four puzzle instances, the TOIDA* algorithm identified a suboptimal state transformation evaluation order that required an average of 25% more node expansions than required by the optimal state transformation evaluation order. When all eight puzzle instances are considered, the extra work required drops to 18%. The test also showed that the work required for a suboptimal evaluation order can be 900% more than that required by the optimal evaluation order. The results of this second test are shown in Table 2.

Puzzle #	TOIDA* expansions	Korf IDA* order Expansions	Improvement
100	68750001	67880056	-1.01
99	170901508	83477694	-2.05
19	1373048	1280495	-1.07
47	1523089	1411294	-1.08
93	5627965	1599909	-3.52
90	9191415	7171137	-1.28
18	41202949	23711067	-1.74
97	950913	1002927	1.05
94	136943	1337340	9.77
85	2461544	2725456	1.11
86	1126101	2304426	2.05
83	45983903	65533432	1.43
28	5890673	5934442	1.01
16	5835264	17984051	3.08
23	15118399	15971319	1.06
74	1897728	2289588	1.21
79	411617	540860	1.31
12	166664	546344	3.28
42	225515	877823	3.89
55	397057	927212	2.34
Average Improvement by case			1.04

Table 3: 20 puzzle instances

Now in Table 3, we show results from 21 initial puzzle configurations. Here we only compare node expansions against the fixed order that Korf used, primarily because some of the puzzles will require prohibitively many node expansions to test all possible fixed node orderings. The time overhead of TOIDA* is being ignored since it appears to be a non-factor in the comparison. One must remember that the one fixed ordering used by IDA* may be the optimal ordering for the problems in which it outperforms TOIDA*. Overall, TOIDA* expands about 96% of the nodes that the fixed order IDA* expands on the 21 examples shown. It performs badly (in comparison) on a couple of the larger problems. Below, we examine its performance on one of these.

On problem 93, TOIDA* is about three times worse in the number of node expansions. Problem 93 has between 1.5 and 5.7 million node expansions or state transformations for the two orderings which takes hours of cpu time on the Intel Hypercube 2/386. We did, however, examine all 24 possible node orderings for problem 93. It was found the the fixed order tested against was within 99.96% of optimal. The worst case for problem 93 was 8,490,547 expansions which was 1.5 times more than the order chosen by TOIDA*

required. The average over the 24 possible orders was 5,044,871 expansions or 12% less expansions than TOIDA* incurred. The point of this analysis is that TOIDA* was only slightly worse than average in this case and significantly better than the worst fixed order in number of nodes expanded.

5. EVALUATION OF TOIDA*

The TOIDA* algorithm can correctly identify a “good” transformation evaluation order with minimal time penalty. The IDA* algorithm can provide more efficient search performances if the optimal state transformation evaluation order is known prior to the search, but if the optimal order is not known the TOIDA* algorithm has been shown to provide a solution that is (in general) much more efficient than the average performance of the IDA* algorithm with all evaluation orders. The TOIDA* algorithm is therefore likely to be more efficient than the IDA* algorithm with an arbitrary state transformation evaluation order for an arbitrary problem. Further, TOIDA* requires that minimal information be saved during an iteration. It only needs a minimum h value and a relative ordering to be used with the next depth limit. Hence, the space and time overhead of using TOIDA* are minimal, making it a good candidate search algorithm when memory resources are limited.

6. SUMMARY

In this paper we present a Transformation-Ordering Iterative-Deepening A* (TOIDA*) search that improves the performance of IDA* by dynamically improving the search order. This results in an improved efficiency of the final search iteration. We describe an implementation of the technique, and present the results of a series of tests performed on the Fifteen puzzle problem. The results are compared with IDA* in terms of search time and number of node expansions, and TOIDA* is shown to outperform IDA* in many of the tests.

Improving the computational cost of heuristic search is an active area of research in Artificial Intelligence, because search dominates many AI algorithms. We have shown that it is possible to improve the performance of IDA* search *without* requiring a substantial increase in storage space, by improving the fixed node expansion order on each iteration through the search space. The research we present in this paper opens up a great many areas of continuing work, which we intend to pursue. These areas include investigating and combining parallel implementations of TOIDA*, analyzing the potential gain of transformation ordering, discovering the optimal fixed search order, and extending the algorithm to save portions of the frontier space.

ACKNOWLEDGEMENTS

This research was partially supported by grant CDA8920890 from the National Science Foundation and by a grant from the Florida High Technology and Industry Council Software Section.

References

- [1] P. P. Chakrabarti, S. Ghose, A. Acharya, and S. C. de Sarkar. Heuristic search in restricted memory. *Artificial Intelligence*, 41(2):197–221, 1989.
- [2] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- [3] R. E. Korf. Optimal path-finding algorithms. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, pages 223–267. Springer-Verlag, 1988.
- [4] V. Kumar and V. N. Rao. Scalable parallel formulations of depth-first search. In Kumar, Kanal, and Gopalakrishnan, editors, *Parallel Algorithms for Machine Intelligence and Vision*, pages 1–41. Springer-Verlag, 1990.
- [5] C. Powely and R. E. Korf. Single agent parallel window search: A summary of results. In *International Joint Conference on Artificial Intelligence*, pages 36–41, 1989.
- [6] C. Powley, C. Ferguson, and R. E. Korf. Parallel heuristic search: Two approaches. In *Parallel Algorithms for Machine Intelligence and Vision*, pages 42–65. Springer-Verlag, 1990.
- [7] V. N. Rao, V. Kumar, and K. Ramesh. A parallel implementation of Iterative-Deepening-A*. In *Proceedings of AAAI-87*, pages 178–182, 1987.

AN IMPROVEMENT OF WEIGHTING STRATEGY IN RESOLUTION-BASED AUTOMATED REASONING

Yong-Gi KIM and Ladislav J. KOHOUT

The Center for Expert Systems and Robotics
Department of Computer Science
Florida State University
Tallahassee, Florida 32306, USA.

Abstract

The fast fuzzy algorithm is used in resolution based automated reasoning to produce the weights of terms automatically. By using the weights extracted by the fast fuzzy algorithm, some of the fuzzy implication operators were compared.

1. INTRODUCTION

Resolution theorem prover systems form an important category of logical architectures in the field of Automated Reasoning. In this paper we outline a method for control of inferential strategies of resolution based architectures which employs the triangle fuzzy relational products and fast fuzzy relational algorithms. The method for speeding up the logical inference is tested in conjunction with the theorem prover called ITP.

ITP has been one of the most important systems in the field [1], developed by Aragonne Laboratory. The Aragonne group used ITP extensively in ATP research [2], proving many theorems, using it to verify software and hardware, solving algebraic word problems as well as some other open mathematical problems. The ITP was distributed over 200 sites, and used extensively by other workers as publications in the Journal of Automated Reasoning indicate. Boyer used this system for proving some basic mathematical theorems in Gödel's axiomatization of set theory [3].

In our approach, ITP is used as the basic architecture, embedded in a many-valued logic based system which controls the selection and priorities the inference strategies by means of many-valued and fuzzy logics based relational algorithms [4] and heuristics, respectively.

After surveying the structure and functional activity of the ITP in section 2 and 3, a new strategy using fuzzy preorder relations is proposed. The preliminary

experimental results and comparison of a whole spectrum of fuzzy implication operators is presented in the sections 5 and 6, respectively.

2. THE USE OF STRATEGIES IN AUTOMATED REASONING SCHEMES

Current automated theorem prover ITP adopts strategies called *set of support* strategy and *weighting strategy*. The reason for using strategies is that the automated reasoning program can avoid many fruitless paths by their judicious and "informed" application. Without a suitable strategy guiding the inference, too many often irrelevant clauses may derive, and those clauses may lead the program easily into a blind alley. Therefore, the strategies are the must in any serious use of automated reasoning. The set of support strategy is one of the most powerful restriction strategies in the resolution-based automated inference systems. The set of support strategy forbids a reasoning program from applying an inference rule unless at least one of the potential parents to which it is being applied has been deduced from some specified subset of the input clauses. Even though the set of support strategy eliminates many fruitless clauses from the inference stream, it is often not powerful enough to produce the conclusion in acceptable time. Hence, weighting strategy is being used with the set of support strategy, in the current theorem prover ITP in addition to the set of support. The weighting strategy assigns some priorities to each term, literal, and clause. With the weighting, one can assist the reasoning program by contributing some of one's own experience capturing one's intuition, in order to give the program hints. Weighting means assigning "weights" to various concepts. The lighter the weight is, the sooner the program will look at the clause. Unfortunately, this weighting strategy is too heuristic and too dependable on the subjective side of one's experience or intuition.

Here, we propose to apply the fast fuzzy relational algorithms [4] as an automatic technique for extracting the weighting strategy. Instead of determining weighting patterns heuristically by an trial and error approach, the new scheme provides for selecting the weights automatically [5], thus replacing by a fuzzy algorithm, the manual selections that have been previously done by the users of the ITP heuristically.

3. THE GLOBAL ACTIVITY OF THE ITP

In order to elucidate the new scheme further, we have to have a closer look at the global activity of ITP. The user of ITP enters theorems which are changed into clause form to ITP. The clauses consist of four clauses which are the axioms list, the set of support list, the have-been-given list, and the demodulator list. The fundamental operation consists of the following steps:

1. Choose a clause from the set of support list. Call this clause "*the given clause*".
2. Infer a set of clauses that have the *given clause* as one premiss; as other premisses the clauses are selected from the axioms list, the have-been-given list, and the demodulator list, depending on the chosen type of inference process.
3. For each generated clause, "process" it (i.e., simplify it, perform subsumption checks, etc.).
4. Move the given clause from the set of support list to the have-been-given list.

The operation of the ITP consists of repeated execution of these four steps until either the set of support list has become exhausted or a contradiction has been found. The user enters selections (i.e. which inference rule should be used and how the given clause is picked up) into the ITP with the clauses of the axioms defining the problem. The user-controlled options govern the step 1, 2, and 3 above. These user-controlled options include selection of the inference rules (e.g. binary resolution, unit resolution, and hyperresolution, etc.) and the weighting scheme for each term, literal, and clause. Since too many clauses are generated through the repeated steps, specific weights are assigned to each term, literal. The clause to be picked up first is the one with the lowest weight.

4. REPLACEMENT OF WEIGHTING STRATEGY BY A FUZZY INFORMATION RETRIEVAL SCHEME

The priority of the second premiss, in the activity step 2 of the previous section, is determined by the weight heuristically assigned by the user of the ITP. We replace this heuristic weighting strategy by fast fuzzy relational algorithms. To apply these algorithms to this particular problem domain, we have to determine the semantic conceptual descriptors [6] characterising the actions [7] of the theorem proving strategies. This is achieved by the application of the Fuzzy Information Retrieval (FIR) scheme [8], thus making the value of the assigned priority the function of fuzzy *logical request* and fuzzy *relational request* [9] of FIR. The major advantage of our new scheme is the fact that the order-like relations determining the priority of the clauses selected to be entered into the inferential stream of the ITP can be identified from the experimental data by fast fuzzy relational algorithms [4].

The functional specification of the activity of FIR used to select the relevant clause is as follows [8]:

What is involved in fuzzy Information Retrieval of clauses can be expressed essentially by means of the following four items:

1. A set D of *clauses* d .
2. A set T of *descriptors* t_j (for example properties of clauses).
3. A *clause-descriptor* relation R , which is a fuzzy relation such that

$$R \in \mathcal{R}_{\mathcal{F}}(D \rightarrow T).$$

Then R_{ij} is the degree to which clause d_i is related to descriptor t_j , which can be viewed as the degree of relevance of the features described by descriptor t_j to clause d_i .

The appropriate characteristics of descriptors have to be determined empirically by a series of carefully designed experiments, or from appropriate theoretical considerations. It is clear that the set of relevant properties of the elements involved in these identification experiments is strongly dependent on the mathematical characteristics of the problems presented to the ITP. In the next section we shall describe one method of constructing the relational matrix, relating the axioms describing the problem to be solved by the ITP with the descriptors characterising its axioms. This fuzzy relational matrix is then used by the TRISYS system [4],[10],[11] to extract the descriptor hierarchy. The automatically extracted descriptor hierarchy is consequently used to speed up the inferential process of the ITP.

5. CONSTRUCTION OF A FUZZY MATRIX FOR BUILDING A DESCRIPTOR HIERARCHY

In order to construct a descriptor hierarchy in the problem domain of resolution based automatic reasoning, the automatic reasoning system needs a matrix which consists of clauses and descriptors which describe the properties of the clauses. The matrix is used for constructing the descriptor hierarchy. This hierarchy consists of the descriptors organized in such a way that the highest descriptor is the most relevant to deriving the conclusion (the empty clause), from clauses by means of inference rules such as binary resolution, and hyperresolution, etc.

Our particular application of the fuzzy information retrieval technique uses a relational matrix which conceptually represent a relation from the set of clauses, to the other set, the properties. The set of clauses used in the fuzzy matrix is formed from the logical *axioms* of the problem to be solved, and from the *immediate consequences* of these axioms. Both are unified with the set of support by a suitable unification algorithm. The immediate consequences consist of the very first level *resolvent* that is generated by applying the selected inference rules to the axioms and set of support, by means of *breadth first* search. The set of properties is formed from the terms which appear in the clauses representing the axioms and

the immediate consequences. The fuzzy relational matrix D_{ij} gives the degree of relatedness between the i -th element of the set clauses and the j -th element of the set of properties. The fuzzy degrees are determined by application of the following rules:

1. If the property j is an element of the clause i , then the degree of the relatedness D_{ij} is 1.
2. If the property j is not an element of the clause i , then the degree of the relatedness D_{ij} is 0.
3. If the property j is an element of a subterm (i.e., $g(a)$) of the clause i , then the degree of the relatedness D_{ij} is 0.5.
4. If the property j is an element of a subterm of a subterm (i.e., $g(g(a))$) of the clause i , then the degree of the relatedness D_{ij} is 0.5×0.5 , and so on.
5. If the property j is an element of the clause i , and the property j is an element of a subterm of the clause i at the same time (i.e., $P(a,b,g(a))$), then the degree of the relatedness D_{ij} will be bigger one applied to the case using the above steps 1 to 4.

6. EFFECT OF THE NEW WEIGHTING TECHNIQUE BASED ON FUZZY DESCRIPTOR HIERARCHY

Our new weighting technique has been applied to several classes of problems that are amenable to the automated reasoning approach. The first example illustrating the results of the new weighting technique to be described here is the following theorem of the group theory: *"In a group, if the square of every element is the identity, the group is commutative"*. Selecting the hyperresolution as the inference rule of the ITP, the default weighting (which assigns no specific weight for each variable but assigns value 1 to each variable by default) produced the conclusion in 146 seconds in 121 inference steps. On the other hand, the new fuzzy method, which used the weighting pattern derived as described above, reached the conclusion in 52 seconds with 49 deduced steps.

The second application listed in the comparison below, involves a digital circuit problem of verifying the correct function of the full adder [12]. The result of the test of the condition *when the inputs to the circuit are high, low, and high, then the outputs levels of the device are low and high* is presented below. Using hyperresolution and UR-resolution simultaneously as the inference rules, the weighting pattern derived by the new technique yields the conclusion in 21 seconds with 49 deduced steps while the default weighting get the empty clause in 26 seconds with 72 steps.

The third demonstration of our technique presented here is concerned with the well known AI problem called "the block problem [12],[13]. In our comparison below we present the microworld of three blocks, in the following arrangement: *The initial state of the block world problem is that block A is on block B, and block B is block C, and the goal state is that the block A is on the table.* Green's method [12], which is a planning procedure based on resolution, was used to solve the block world problem. Using the hyperresolution as the only inference rule, the ITP obtained the conclusion in 55 seconds in 119 steps when using the breadth first search method; in 22 seconds in 65 inference steps while using the default weighting pattern, and in 15 seconds in 37 steps by using the weighting pattern derived by our new fuzzy technique.

The new weighting pattern derived automatically by the fast fuzzy relation algorithms encapsulated in the TRISYS system [4] was tested in preliminary experiments significantly reduced the total number of steps and CPU time that the ITP needed to reach the conclusion. The performance results of the solution to all the three problems just described are listed in the table below. The experiments were run on SUN 3/50 under UNIX, where the ITP was installed.

	group theorem	digital circuit	block world
inference rule	hyperresolution	hyperresolution & UR-resolution	hyperresolution
default weighting	121 steps 146 seconds	72 steps 26 seconds	65 steps 22 seconds
weighting of new tech.	49 steps 52 seconds	49 steps 21 seconds	37 steps 15 seconds

The Hasse diagrams extracted by TRISYS that provided the weighting leading to the results listed in the above table were computed by the algorithm described in [4] using the implication operators as follows: group theorem - L5 (α -cut at .89, mean level); digital circuit - L6 (α -cut at .93, half-upper level); block world - L55 (α -cut at .95, half-upper level);

7. COMPARISON OF RESULTS USING VARIOUS IMPLICATION OPERATORS

To investigate the effect of various many-valued logic operators on the process of extractions of the weighting hierarchies derived by the fast fuzzy relational algorithms

two of the three problems described in the previous section were selected for further study:

- The theorem of the group theory.
- The problem of the verification of the digital circuit called “full adder”.

The extracted weighting hierarchies and number of steps and CPU time to get the conclusion are given below. Eleven different operators were compared for each problem. Among them, selected to show the effect are L5, L55, and L6. L5 is *Lukasiewicz*, L55 is *Kleene-Dienes Lukasiewicz* and L6 is *Kleene-Dienes*, respectively.

Shown in the tables below is the number of steps which it took to produce the conclusion, followed by the numbers in parenthesis which represent the CPU time taken to reach the conclusion.

1. The theorem of group theory:

2. The logical circuit:

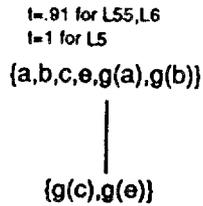
α -cut	L5	L55	L6	L5	L55	L6
1	112 (4:18)	112 (4:18)	112 (4:18)	80 (:21)	98 (:28)	98 (:28)
.95	50 (1:01)	50 (1:01)	50 (1:01)	87 (:23)	85 (:23)	78 (:20)
.9	49 (:54)	49 (:54)	49 (:54)	107 (:32)	105 (:31)	105 (:31)
.85	49 (:54)	49 (:54)	49 (:54)	107 (:32)	100 (:29)	105 (:31)

The inference rules were *hyperresolution* and *UR-resolution*, used simultaneously. For comparison, the default weighting produced the conclusion (the empty clause) in 2 minutes 27 seconds, in **83** steps for the group theorem, and in 28 seconds in **98** inference steps for the logic circuit, respectively.

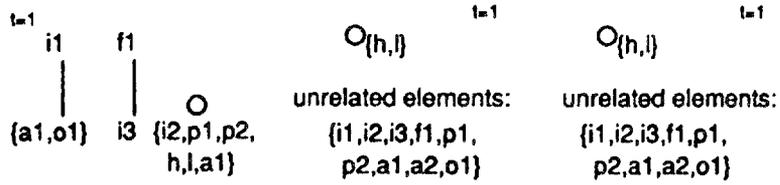
The Figure shows some Hasse diagrams, depicting the automatically obtained weighting hierarchies extracted by TRISYS, based on methods described in Sec.4 and 5 above. Note the differences in Hasse diagrams for different knowledge domains to which the theorem prover is applied. The lighter the weight is, the sooner the mechanical theorem prover picks the clause. Therefore, the term located at the highest in the descriptor hierarchy will be assigned a lightest weight.

inference rules: hyperresolution + UR-resolution

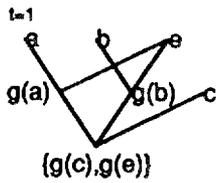
alpha-cut: 1



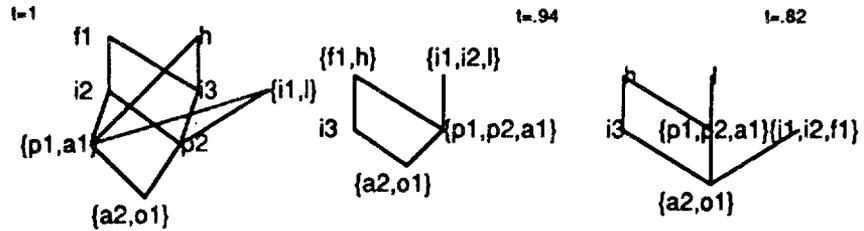
alpha-cut: 1



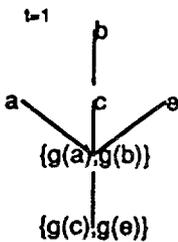
alpha-cut: .95



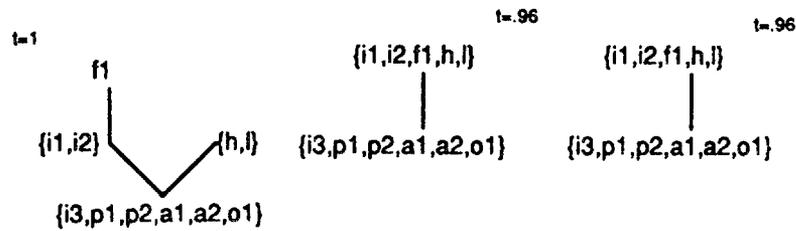
alpha-cut: .95



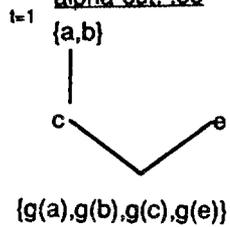
alpha-cut: .90



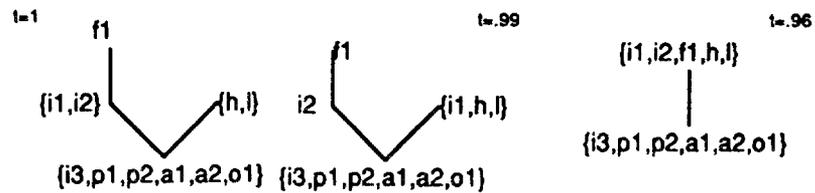
alpha-cut: .90



alpha-cut: .85



alpha-cut: .85



L5,L55,L6

m=.81
sd=.195

L5

m=.89
sd=.091

L55

m=.88
sd=.101

L6

m=.87
sd=.106

hierarchies of the math theorem

hierarchies of the logic circuit - full adder.

REFERENCES

1. Lusk, E., and Overbeek, R., "The automated reasoning system ITP," Technical Report ANL-84-27, Argonne National Laboratory, Argonne, IL, 1984.
2. Wos, L., Overbeek, R., Lusk, E., and Boyle, J., "*Automated Reasoning: Introduction and Applications*", Prentice Hall, Englewood Cliffs, NJ, 1984.
3. Boyer, R. S. et al., "Set theory for first order logic: Clauses for Gödel axioms," in *Journal of Automated Reasoning*, 1986.
4. Bandler, W. and Kohout, L.J., "Special properties, closures, and interiors of crisp and fuzzy relations," in *Fuzzy Sets and Systems*, pages 26(3):317-332, June, 1988.
5. Kohout, L.J., and Kim, Yong-Gi, "Use of fuzzy relational information retrieval techniques for generating control strategies in resolution-based automated reasoning," in "*FLAIRS - 90 Proceedings*, The Florida Artificial Intelligence Research Society, pages 140-144, 1990.
6. Kohout, L.J., "Activity Structures: a methodology for design of multi-environment and multi-context knowledge-based systems," in Kohout, L.J., Anderson, J., and Bandler, W., editors, "*Multi-Environmental Knowledge-Based Systems*," chapter 5, Gower, Aldershot, U.K., 1991.
7. Kohout, L.J., "*A perspective on Intelligent Systems: A Framework for Analysis and Design*, Chapman and Hall & Van Nostrand, London & New York, 1991.
8. Kohout, L.J., and Bandler, W., "The use of fuzzy information retrieval technique in construction of multi center knowledge based systems", in Bouchon, B., and Yager, R.R., editors, *Uncertainty in Knowledge-Based Systems*, pages 257-264, Springer-Verlag, Berlin, 1987.
9. Kohout, L.J., Keravnou, E., and Bandler, W., "Automatic documentary information retrieval by means of fuzzy relational products", in Gaines, B.R., Zadeh, L.A., and Zimmermann, H.-J., editors, *Fuzzy Sets in Decision Analysis*, pages 308-404, North-Holland, Amsterdam, 1984.
10. Kohout, L.J., and Bandler, W., editors, "*Knowledge Representation in Medicine and Clinical Behavioural Science*," Abacus Book, Gordon and Breach, London and New York, 1986.
11. Bandler, W. and Kohout, L.J., "Hierarchies in Symptoms and Patients through Computation of Fuzzy triangle products and closures," in Parslow R.D. ed. *BSC'81 Information Technology for the Eighties*, Heyden & Son LTD., 1981.

12. Genesereth, M., and Nilsson, N., *Logical Foundations of Artificial Intelligence*,
” Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.
13. Shirai Y., and Tsujii, J., *Artificial Intelligence: concepts, techniques, and ap-
plications*, John Wiley & Sons, 1984.

**PATTERN RECOGNITION USING
THE THIRD AND THE FIFTH CLASSES OF DYNAMICAL SYSTEMS**

Ying Liu

Hede Ma

Dept. of Math. and Comp. Sci. Dept. of Eng. Tech
School of Sciences and Technology
University System of Georgia
Savannah State College
Savannah, Georgia 31404

ABSTRACT

In this paper, dynamical systems are classified into eight classes by the authors. Thereafter, efficient methods are developed in this paper to recognize an input vector using the third and the fifth classes of dynamical systems defined by the authors. Patterns classified and recognized by a dynamical system are based on attractors of the system. Wolfram's cellular automata, Barnsley's iterated function systems (IFS), and ω -Orbit Finite automata proposed by the authors are used to illustrate the idea that how patterns can be recognized by the third and the fifth classes of dynamical systems, which is a more efficient approach than the existing methods.

Key words: Iterated functions systems, finite automata, attractor, fractal, cellular automata, configuration space, pattern recognition, neural network, dynamical systems, ω -orbit finite automata.

1. INTRODUCTION

Because of the computation complexity, it is difficult to recognize arbitrary patterns at a reasonable cost. Many problems of how to recognize patterns efficiently remain open. In a simplified approach to pattern recognition, a pattern recognizer operates as a "black box" which receives an input vector x and produces a response η_i on one of its output ports i , each port being assigned to a different class of observed items [1]. If x belongs to class i , $\eta_i = 1$ and $\eta_j = 0$ for all j not equal to i .

The theoretical problem is to devise an adaptive process such that given a set of input pairs $\{ (x_k, \eta_k), k = 1, 2, \dots, K \}$, one finally hope to obtain a recognizer that will recognize all the input vectors.

Neural network models of the Hopfield type have drawn intensive attention in the past years mainly because of their capacities as associative memory and fast computing device [1,2,3]. When an unknown pattern is imposed on a Hopfield net, the net iterates in discrete time steps using a given formula. The net is considered to have converged when outputs no longer change on successive iterations. This fixed pattern determines which class the unknown pattern belongs to.

A network can be operated in two different modes. Let a net has N neurons. In synchronous operation, each of the N neurons simultaneously evaluates and updates its state according to a rule. In asynchronous operation, the components of the current state x are updated one at a time according to a rule. Hopfield and his colleague [2] proved that a net evolves toward a stable configuration if the mapping rules specified by a matrix are symmetric and the mode of operation is asynchronous.

The Hopfield model has several unsolved problems as follows, which motivates our research work to establish more efficient methods in pattern recognition.

(1) No theory is available to specify the conditions under which a net evolves toward a stable configuration if the mode of operation is synchronous (the synchronous mode is more favorable because of fast convergence).

(2) The information capacity of the Hopfield nets is limited. The number of arbitrary vectors that can be made stable in a Hopfield network of N neurons is proved to be $O(N)$ [3]. It is very expensive and unattractive to use N neurons for storing $O(N)$ patterns.

(3) In order to specify a system with N -bits configurations, $O(N^2)$ real parameters must be used (16 - 32 bits for each real number). With introducing more parameters ($O(N^2)$) to specify fewer bits ($O(N)$), it is expected that a direct construction of a net from a given set of input pairs can be achieved. Therefore, the time complexity of neural network inference could be reduced. One of such approach is the outer product construction of a Hopfield net from a given set of input pairs [2]. But such a direct inference has not been achieved for synchronous net. If no such direct inference algorithm can be found, it is necessary to introduce some search algorithms in the parameter space ($O(N^2)$ dimensional) for inference of a net from a given set of input pairs. The parameter space for a neural network has high dimensions. Consequently, the time complexity for inference is high.

The problems listed above motivate us to develop some new approaches. We have extended the Hopfield models in two directions (both use dynamical systems [5,10]). One of the two direction is presented in this paper, and the other one will be presented in [4].

In this paper, efficient methods are developed to recognize an input vector using the third and the fifth classes of dynamical systems defined in this paper. Patterns classified and recognized by a dynamical system is based on attractors of the system. Wolfram's Cellular Automata (CA) [10], Barnsley's Iterated Function Systems (IFS) [5], and ω -Orbit Finite Automata (ω -OFA) [9] proposed by authors are special cases of dynamical systems. They can be used as models for implementing a pattern recognition procedure.

In Hopfield model [1,2,3], a net has a memory. Given a set of input pairs which is used to train the net, the input vectors are stored in the memory. Vectors that are in the memory are also fixed point of the neural net. These fixed points exercise a region of influence around them. Configurations which are sufficiently similar to a fixed point are mapped to the memory by repeated iterations of the system operation. In other words, input vectors that are used to train a net are treated as fixed points or point attractors of the net.

Our scheme can be considered as a generalization of the Hopfield model, which will be more powerful and efficient in pattern recognition. An input vector x is recognized according to its attractor. These attractor can be cyclic attractors instead of fixed points. For the case of fixed points, these attractors are in general not the same as the set

of the input vectors which are used to train the net. In the set of training input pairs, many different input vectors can be in the same class. For a trained dynamical system, let x be an input vector, if the evolution of a dynamical system with the initial configuration x leads to an attractor i , x belongs to class i .

In this paper, section 2 introduces several basic concepts. Section 3 introduces our classifications of dynamical systems. In section 4, we explain our idea in detail. In section 5, Wolfram's cellular automata are used to implement our idea. In section 6 and 7, Barnsley's iterated function system [5] and the ω -Orbit Finite Automata proposed by the authors used for the same purpose. Finally, a conclusion is drawn in section 8.

2. DEFINITIONS

In this section, we introduce several basic definitions.

Definition 1

Let X be a complete metric space. Then the set of all compact subsets of X except the empty set is denoted as $H(X)$ [5].

Definition 2

A dynamical system consists of a configuration space $G(X) \subset H(X)$ together with a mapping $F: H(X) \rightarrow H(X)$. $G(X)$ is closed under F . Formally,

$$D = (G(X), F).$$

Definition 3

Let $D = (G(X), F)$ be a dynamical system. Let

$$\begin{aligned} \Omega^{(0)} &= G(X) \\ \Omega^{(1)} &= F(\Omega^{(0)}) \\ \Omega^{(n)} &= F(\Omega^{(n-1)}), \end{aligned}$$

then

$$\Omega = \bigcap_{i=0}^{\infty} \Omega^{(i)}$$

is called an attractor set of D [5,10].

Definition 4

A null attractor p_0 is the empty set [10];

A point attractor p_1 is a configuration such that

$$F(p_1) = p_1;$$

A cyclic attractor is a set of configurations

$$Q_k = \{ p_{k1}, p_{k2}, \dots, p_{kk} \mid F(p_{ki}) = p_{k\ i+1}, \quad F^{(k)}(p_{ki}) = p_{ki}$$

A regular infinity attractor is a cyclic attractor where k is infinity and the set can be specified by a finite amount of information;

A strange attractor is the one which has an infinity number of

configurations and which can not be specified by a finite amount of information.

Definition 5

Let (X,d) be a complete metric space. Then the Hausdorff distance [5] between points A and B in $H(X)$ is defined by

$$\begin{aligned} h(A,B) &= \text{Max}\{ d(A,B), d(B,A) \} \\ d(A,B) &= \text{Max}\{ d(x,B) : x \in A \} \\ d(x,B) &= \text{Min}\{ d(x,y) : y \in B \}. \end{aligned}$$

Definition 6

A transformation $f : X \rightarrow X$ is called a contraction mapping on $[a,b]$ if

- (1) $f(x)$ is continuous on $[a,b]$;
- (2) $f(x)$ is in $[a,b]$ for all x in $[a,b]$;
- (3) $|f'(x)| \leq s < 1, \quad x \in [a,b]$

where $|a|$ means absolute value. The minimum number of such s is called a contractive factor for $f(x)$.

Definition 7

An Iterated Function Systems (IFS) [5] consists of a complete metric space (X,d) together with a finite set of contractive mapping Σ [5]. Formally, an IFS, A , is written as

$$B = \{ X, \Sigma \}, \quad \Sigma = \{ w_0, w_1, \dots, w_{k-1} \}.$$

Here w 's are contractive mappings.

Definition 8

An ω -Orbit Finite Automata (ω -OFA), A , is a 5-tuple [9]

$$A = \{ R; \Sigma; M; I; F \}$$

where (1) R is a finite set of states

$$R = \{ X_1, X_2, \dots, X_i, \dots, X_n \}$$

and X_i is a metric space. (2) Σ is an affine alphabet, which is similar to an IFS transformation alphabet. (3) M is an $(n \times n)$ transition matrix. Each elements of this matrix, M_{ij} , is a subset of Σ . The transition rule is

$$(X_j, w) \rightarrow X_i, \quad \text{if } w \in M_{ij}.$$

The set M_{ij} can be empty. (4) The initial state is $I = \{ X_1 \}$, and (5) the final states F is a subset of R .

An ω -OFA, A , accepts an ω -regular language $L(A)$ [6,7]. The alphabet of $L(A)$ is a set of affine transformations Σ . $L(A)$ is also defined as the orbit language of ω -OFA: $O(A) = L(A)$. The images defined by ω -OFA are

$$A_{\omega} = L(A)X_0 = O(A)X_0, \quad X_0 \in H(X).$$

where

$$L(A)X_0 = \{ x \mid x = ux_0, \quad x_0 \in X, \quad u \in L(A) \}.$$

As a special example, an IFS $A = \{X, \Sigma\}$ is an one-state ω -OFA

$$A = \{ X; \quad \Sigma; \quad M_{1,1} = \Sigma; \quad X; \quad X \}.$$

Hence the orbit language of an IFS is

$$O(A) = L(A) = \Sigma^{\omega}.$$

Definition 9

A binary CA [10] is $C = \{X, \Sigma\}$, where X is a finite or infinity array and Σ is a mapping rule. For one dimensional binary CA,

$$\Sigma = w_1 = \{ r, R, k \}.$$

(1) $k = 2$ and each site value is specified in the range 0 through $k-1$. (2) The site values evolve by iteration of the mapping

$$a_i^{(t)} = f_R (a_{i-r}^{(t-1)}, \dots, a_{i+r}^{(t-1)}),$$

and (3)

$$R \in \{ 2, 4, \dots, 2^{2r+1} \}$$

is the code number for a mapping.

For example, $\{r, R, k\} = \{2, 20, 2\}$ means site value = $\{ 0, 1 \}$, neighborhood size is $2r+1=5$, and the CA uses rule 20.

3. CLASSIFICATIONS OF DYNAMICAL SYSTEMS

First of all, a dynamical system is described by a state $x \in X$ of the system. Such a state x is also called a configuration of the system [10]. All such configurations together form a space $H(X)$, called a configuration space [10]. Some dynamical systems only use a subspace of $H(X)$. Secondly, an evolution of a system in its configuration space is specified by a set of production rules [10]. Examples of discrete dynamical systems are cellular automata (CA) [10], and examples of continuous dynamical systems are iterated function systems (IFS) [5]. The advantage of CA is that the mapping rule of a system in its configuration space is determined locally [10]. Therefore, CA can be used to simulate a large variety of natural phenomena. The advantage of IFS is that the mapping of a system in its configuration space is continuous. Therefore, mathematically, IFS can be

handled easily.

A specification of a dynamical system includes a definition of a configuration space and a set of rules for motions in the configuration space [5,10]. Most of dynamical system evolutions are irreversible [10]. An orbit of a system is a trajectory in its configuration space. Orbits merge with time, and after many time steps, orbits starting all possible initial configuration become concentrated onto "attractors" [5,10]. These attractors typically contain only a very small fraction of possible configurations. Evaluation to attractor from an arbitrary initial configuration allows for pattern recognition behaviors.

Dynamical systems can be classified by its attractors [10]. Wolfram classified dynamical system into four classes[10]. In this section, we first define 8 classes of dynamical systems. Then we classify dynamical systems into 8 classes, from the graph topology of possible orbits of a system in its configuration space.

Definition: Let $D = \{ G(X); F \}$ be a dynamical system, where X has infinite elements. A dynamical system D is said to be in

class 1 if it only has a null attractor

$$\Omega_1 = \{ p_0 \};$$

class 2 if it only has a point attractor

$$\Omega_2 = \{ p_1 \};$$

class 3 if it has more than one but a finite number of point attractors

$$\Omega_3 = \{ p_1, p_2, \dots p_K \};$$

class 4 if it has an infinite number of point attractors;

$$\Omega_4 = \{ p_1, p_2, \dots \dots \};$$

class 5 if it has a finite number of cyclic attractors;

$$\Omega_5 = \{ q_{a1}, q_{a1}, \dots q_{ak} \};$$

class 6 if it has an infinite number of cyclic attractors;

class 7 if it has at least one regular infinite attractors;

class 8 if it has at least one strange attractors.

For a dynamical system where X is a finite set, then D must be in class 1, 2, 3, or 5. However, we can extend $D=(G(X),F)$ to $D'=(G(X'),F)$ where X' is an infinity set. Then the classification of D and D' might be different.

Examples of the class 2 dynamical systems are IFS[5] and ω -OFA[9]. Examples of the class 3 dynamical systems are asynchronous neural networks[1,2]. Many CA and synchronous neural nets are in class 5.

Theorem 1: A dynamical systems must be in one of the eight classes.

This theorem can be proved by the topology of orbits of a dynamical system in its configuration space. Since space are limited, the proof of the above theorem will be omitted. If a system has more than one attractors, some of them might not be stable. The null attractor and the single point attractor will always be stable.

Different attractor systems can serve for different purposes. In [4], we are interested in the single point attractor system, i.e. the second class of dynamical system because the single point attractor will be used as a container for input vectors. Here we interested the third and the fifth classes for efficient pattern recognition, which is discussed in detail in the next section.

4. AN EFFICIENT SCHEME FOR PATTERN RECOGNITION

In this section, we first show the current version of neural net might not be efficient because of its limitation on information capacity. Then we present our approach. Finally, we compare our approach and the Hopfield model.

4.1 INFORMATION CAPACITY OF THE HOPFIELD MODEL

We first show that in general, it is impossible to construct a neural net from a given set of training input pairs. To proof this, consider the information capacity theorem of Abu-Mostafa and Jaques [3]. The number of arbitrary configurations that can be made stable in a Hopfield net with N neurons is up Bounded by N [3]. Let a training set of input pairs contain more than N classes of patterns, then it is simply impossible to infer a net to recognize all the input vectors. An extended neural net might escape this limit [12]. If the storing capacity is $O(N)$, the relative storing capacity is

$$\frac{O(N)}{2^N} \rightarrow 0, \quad N \rightarrow \infty$$

which shows that the Hopfield neural net is not efficient.

4.2 PATTERN RECOGNITION USING CLASS 3 AND 5 DYNAMICAL SYSTEMS

We now present our approach. Note that both class 3 and class 5 dynamical systems have a finite number of attractors. Given a pattern, that is, given an input vector x where the size of x is typically between one million to one billion, we can treat x as an initial configuration of a third or a fifth dynamical system. The dynamical system subsequently evolves to its attractor. Let a system have a finite number $K > 1$ of attractors, if the evolution of the dynamical system with the initial configuration x leads to an attractor p_{Ki} , where i is in $\{1, 2, 3, \dots, K\}$, x belongs to the class i . Since the system only has K attractors, all the possible input vectors are classified into K classes.

The pattern recognition problem using the third and the fifth class dynamical systems is defined as follows: to devise an adaptive process such that given a set of pairs of input and output, one finally hopes to recognize all the input vectors by choosing a proper dynamical system, that is, by choosing a proper configuration space together with a mapping rule. We also call this problem as an inference problem. In a simplest

approach, the configuration space is determined by the input vectors. Therefore, the inference problem is to find a mapping rule such that the inferred dynamical system can recognize all the input vectors. There are examples where the configuration space are different from the input image space. we will discuss this in [12].

4.3 A COMPARISON OF OUR APPROACH AND HOPFIELD MODEL

The difference between our approach and the Hopfield model are given as follows. (1) In the Hopfield approach, the fixed points are the input patterns used to train the net. In our approach where the third class dynamical systems are used, the input pattern x_i evolves to an attractor p_{ki} , where x_i and p_{ki} are different. The adaptive process is to train a dynamical system from a set of input pairs. The trained system will has a set of attractor $\{p_{ki}, i=1,2,\dots,K\}$. The system with the initial state x_i will lead to the attractor p_{ki} . Therefore, this approach is a generalization of the Hopfield model. (2) In our approach where the fifth class dynamical system is used, i.e., cyclic attractors are used, no similar approach can be found in the Hopfield model. This approach covers a larger class of dynamical systems. (3) There is no $O(N)$ information storing limit for our approach. Therefore. it is more powerful and efficient.

5. CA APPROACH

CA are discrete dynamical systems with simple construction but complex self-organizing behavior [10]. They are mathematical models for complex natural systems containing a large number of simple identical components with local interaction. This structure is specially favored by massive parallel computation. CA consist of a lattice of sites, each with a finite of possible values. The values of the sites evolve synchronously in discrete time steps according to identical rules. The value of a particular site is determined by the previous values of a neighborhood of sites around it. The hardware implementation of CA constructs a special type of systolic arrays.

A CA, $C = \{ X, \sum \}$, with a finite X can be in the third or the fifth class using our definition. Therefore, our ideas can be applied by choosing CA as dynamical systems. In the following, we first present a fifth class CA. Then we show how the fifth class dynamical system is used in pattern recognition. Finally, a comparison of this approach with neural net is given.

5.1 AN EXAMPLE OF CLASS 5 DYNAMICAL SYSTEM

Let a CA be specified by

1. $X = \{ 0, 1, 2, \dots, N-1 \}$, $N = 100$, i.e. we have an one-dimensional ring CA of size 100;
2. $\{ R, r, K \} = \{ 2, 20, 2 \}$, i.e.
 - $k = 2$, i.e. site values can be either 0 or 1;
 - $r = 2$, i.e. the site value of the i 'th site is determined by the previous site values of sites $i-2, i-1, i, i+1, i+2$, in MOD N ;
 - code number = 20, i.e. if the sum of previous site values of the sites $i-2, i-1, i, i+1$ and $i+2$ is 2 or 4, the site value of the site i is 1. Otherwise, it is 0.

If we omit the translational invariance, repetition, and combinations (See the explanation below), this CA has 12 attractor atoms [10]. These attractors are represented by digital numbers:

Period	Minimal configuration in decimal number
1	0 (Null Attractor)
2	151 (00...0010010111)
9R	187 (00...0010111011)
1	189 (00...0010111101)
22	195 (... ..)
9L	221
1R	635
1L	889
38	125231
4	595703
4	610999
4	624623 (00...001001100001111101111)

All other attractors can be made in the following ways:

1. A spacial translation of the above attractors. For example, from the attractor 151, we can generate new attractors 151x2, 151x4:

Period	Configuration
2	151x2 (00...00 100101110)
2	152x4 (00 0 1001011100)

2. A repetition of a above attractors. For example, from 189 one can generate a new attractor:

Period	Configuration
1	198 x(1 + 2 ¹⁰) (00...00101111010010111101)

3. A combination of two or more above attractors. For example, from the attractors 151 and 189, an attractor 189 + 151 x 2¹⁰ can be constructed.

This CA has a finite number of attractors, including a few cyclic attractors. Therefore, it is in the class 5. In the limit where the size of the CA goes to infinity, it is in class 7, because of the attractor 9R, 9L, 1R, and 1L.

5.2 PATTERN RECOGNITION USING CA

Now we apply our idea by using the above CA as a fifth class dynamical system. There are total of $2^{100} - 1$ input vector x in $H(X)$. They are classified according to the attractors. The attractors are labeled by digital numbers. Given an input vector x , if the evolution of the CA with the initial input vector x leads to the attractor 151, x belongs to the class 151. For example,

```
input vector: x = 00...0010111111000
CA evolution: 00...0010110010100
              00...0010010111100
              00...0001110111010
              00...0010011101010
              00...0001000100110
```

```

00...0000010010111 ( 151 )
00...0000001111001
00...0000010010111 ( 151 )
... ..

```

The above input vector x is recognized as in the class 151. The attractor 151 is a cyclic attractor.

5.3 A COMPARISON WITH HOPFIELD MODEL

Compared with Artificial Neural Network (ANN), we observe

(1) No theory is valuable to specify the condition under which a CA of size N is in the third class. The only algorithm for this testing is enumerative search, which is apparently not very practical for large N . This is similar to the situation of synchronous ANN.

(2) We have extended the pattern recognition algorithm to include the fifth class of CA, as seen in the above example. As a result, we expect the information capacity to be increased.

(3) The information capacity of CA is observed to be $O(N)$, which is not better than ANN. However the power of CA can be extended easily to increase its information capacity at a very low cost. The cost to extend ANN is more expensive. We will discuss these extensions in [12].

(4) There has been no intention to directly construct a CA from a given set of input pairs, like the outer product construction of ANN. However, a direct inference of ANN from a given set of input pairs has not been successful so far. Considering a search algorithm, the parameter space for CA is much smaller. For one dimensional case, these parameter spaces are specified by

$$T = \bigcup T_r, \quad T_r = \{r, R\}, \quad R \in \{2, 4, \dots, 2^{2r+1}\}$$

where $r = \{1, 2, \dots, N/2\}$ is the neighborhood size, and R is the rule code. Even the CA is extended to more powerful classes [12], the parameter space is still relatively small, as compared with ANN. Therefore, the inference of a CA from a given set of input pairs is much easier than ANN.

6. PIFS APPROACH

An IFS [5] consists of a complete metric space X together with a mapping rule: $X \rightarrow X$. It has been shown that [5] if the mapping rules are contractive, a single point attractor system is created. In this paper, we study piecewise IFS (PIFS).

6.1 AN EXAMPLE OF CLASS 3 DYNAMICAL SYSTEM

In this section, we extend Barnsley's IFS to piecewise IFS (PIFS). Let $A_1 = \{X_1, W_1\}$ and $A_2 = \{X_2, W_2\}$ be two IFS. Let $A = \{X, W\}$ be a new PIFS constructed such that X is the union of X_1 and X_2 and W is the union of W_1 and W_2 . Then the IFS A has three attractors: the attractor of A_1 , the attractor of A_2 , and the union of A_1 attractor and A_2 attractor. In general, if we compose a Piecewise IFS A from L IFS, the PIFS A has $2^L - 1$ attractors. In the following, we will present an example of PIFS to show how PIFS can be used in pattern recognition.

As an example, let a PIFS be specified as follows

1. $X = [0, 1]$, i.e. the configuration space $H(X)$ is the set of all

compact subsets of X ;
 2. Let the mapping rule be

$$w(y) = y/2, \quad y \text{ in } [0, 0.5);$$

$$w(y) = 1 - y/2, \quad y \text{ in } [0.5, 1].$$

This system has three attractors $\{0\}$, $\{1\}$ and $\{0,1\}$. This PIFS is in the class 3 since it has a finite number of point attractors.

6.2 PATTERN RECOGNITION USING PIFS

In the above example, the PIFS has three attractors which correspond to three classes of image:

- A. $\{ 0 \}$ for all images in $[0,0.5)$;
- B. $\{ 1 \}$ for all images in $[0.5, 1]$;
- C. $\{ 0, 1 \}$ for all images other than class 1 and class 2.

For simplicity, let the size of input vectors be 30. Let an input vector be

$$x = 11111 \ 11111 \ 11110 \ 00000 \ 00000 \ 00000$$

The subsequent evolution will be

input vector	x =	11111	11111	11110	00000	00000	00000
IFS evolution		11111	11000	00000	00000	00000	00000
		11110	00000	00000	00000	00000	00000
		11000	00000	00000	00000	00000	00000
		10000	00000	00000	00000	00000	00000
(attractor)		10000	00000	00000	00000	00000	00000

Therefore this input vector x is recognized as in the class A, because the evolution if the PIFS with initial configuration x hits the attractor $\{0\}$.

7. ω -ORBIT FINITE AUTOMATA (ω -OFA) APPROACH

ω -OFA [8,9], proposed by the authors, is a generalization of IFS which is more powerful than IFS in image generation. An one-state ω -OFA is an IFS. Using a finite automata as control device in an IFS, an ω -OFA is generated. Formally, an ω -OFA is a 5-tuple, just like a finite automata, except its alphabet is a set of transformations [9]. Also, only ω -strings [7] are used to define its attractors.

There are images which can not be produced by IFS [9] but can be produced easily by ω -OFA. ω -OFA can be used in pattern recognition in the same way as IFS. A piecewise ω -OFA can be defined in a similar way as the case of IFS. Therefore, they construct a third class dynamical system and can be used to implement our idea.

8. CONCLUSION

In conclusion, we have suggested that dynamical systems can be used to recognize patterns. We have shown a powerful and efficient approach for pattern recognition using class 3 and class 5 dynamical systems, defined by the authors in this paper. Specially, Wolfram's CA, Barnsley's IFS and ω -OFA introduced by authors has been demonstrated in playing this role in this paper.

In [12], we will apply the above ideas to four other types of dynamical systems: LM0 and LM1 cellular automata, and LM0 and LM1 neural networks. These four type dynamical systems are proposed by the authors. The inference algorithm for dynamical systems from a set of input pairs will be presented in the coming papers. A complete different approach using class 2 dynamical systems will be presented in [4].

REFERENCES

1. T. Kohonen, An introduction to Neural Computing, Neural Networks, Vol.1, pp.3-16, 1988.
R. Lippmann, In introduction to Computing with Neural Nets, IEEE ASSP MAGAZINE, APRIL, 1987, 4-22.
2. J. J. Hopfield, "neural networks and physical systems with emergent collective computation abilities," in Proc. Nat. Academy Sci., USA, vol. 81, pp.3088-92, 1984.
3. Y. S. Abu-Mostafa and J. St. Jaques, " Information capacity of the Hopfield model," IEEE trans. Inform. Theory vol. IT-31, pp.461-464, 1985.
R. J. McEliece, et al., "The capacity of the Hopfield Associative Memory," IEEE Tran. Inform. Theory vol. IT33, pp.461-482, 1987.
4. Ying Liu and Hede ma, "Algorithms for moving object pattern recognition using ω -Orbit Finite Automata," to appear in Proceedings of Simulation Technology Conference International 1991, Orlando, Fl., Oct. 1991.
5. M.F. Barnsley, Fractals Everywhere, Academic Press, 1988.
M.F. Barnsley, The Desktop Fractal Design Handbook, Academic Press, 1989.
M.F. et. al. Barnsley, Construction Approximation, V5, p1, 1989.
6. J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Language, and Computation, Addison-Wesley, 1979.
7. L. Boasson and M. Nivat, Adherence of Languages, Technical Preprint, Universite Paris 7, Uer De Mathematilques, 1979.
8. Ying Liu and Hede Ma, " ω -Orbit Finite automata and Image Compression," Proceeding of 44th Annual Conference, the Society of Imaging Science and technology, St. Paul, Minnesota, May 1991, 489 - 491.
9. Ying Liu and Hede Ma, " ω -Orbit Finite for Data compression," Proceeding of Data Compression Conference '91, IEEE Computer Press, Snowbird, Utah, April, 1991, 165 - 174.
10. S. Wolfram, "Universality and Complexity in Cellular Automata", Physica 10D, (1984) 1-35.
11. Ying Liu, "A parallel algorithm for traversing a tree", Proceedings of the ISMM International Symposium on Industrial, Vehicular and Space Applications of Microcomputers, pp.58 - 59, New York, October 10 - 11, 1990.
12. Ying Liu and Hede Ma, "A parallel pattern recognition algorithm using LM cellular automata and LM neural networks", to appear in Proceedings of Fourth ISMM International Conference on Parallel and Distributed Computing and Systems, Washington, D.C., October 8 - 11, 1991.

SENSORY INTEGRATION

G. T. McKee

Department of Computer Science, University of Reading, England.

&

E. T. Powner

**Department of Electrical Engineering and Electronics,
UMIST, Manchester, England.**

ABSTRACT

This paper is concerned with the integration of sensory data drawn from a heterogeneous set of sensors. A basic architecture for the sensory subsystem of an intelligent machine is developed, single modality and multiple modality sensory information processing are introduced, three levels are identified for sensory integration (single sensor, multiple sensor and multiple modality) and the forms of sensory integration required at each level are introduced and discussed.

1. INTRODUCTION

We are moving slowly but surely towards the age of "multi-sensor" machines. Primitive versions of such machines have been around for some time. Primitive, that is, when compared to human beings, for they do not possess the full complement of sensors possessed by human beings, and the sensors they do possess are only a shadow of their human counterparts. Industrial robots are the typical example. The primitive character of these machines is reflected in the lack of any detailed theory of sensory systems, or indeed of any systematic methodology for engineering their sensory mechanisms. Of course, given their primitive nature, a detailed theoretical understanding is obviously of little importance, and likewise an engineering methodology. However, as technology moves towards more sophisticated sensory machines, and as the demand for the correct and efficient engineering of these machine increases, the need for this theory and methodology becomes both obvious and urgent.

A number of problems need to be addressed when providing machines with a sensory capability. There is, first, the "selection" problem: the problem of determining the set of sensors required by a particular machine. The answer to this question will be determined by the function the machine is to perform and will be explicitly stated in the machine specification. This paper is concerned with the fact that this specification may include sensors of many different types (vision, tactile, pressure, force, sound, and others).

A second problem is the "strategic" problem: How is the sensory data provided by the sensors to be used in problem solving? This problem, like the first, is not the direct concern of this paper. This paper is concerned, rather, with a third problem, the "multi-sensor" problem. This is the problem of processing the sensory data provided by the multitude of sensors possessed by the machine, drawing that raw and/or processed sensory data together within and across many sensors, and transforming it into a form suitable for use by the problem solving

mechanisms of the machine. In this paper a Sensory Systems Theory is posed as the proper solution to this problem.

Given that one wishes to configure a machine with a specified complement of sensors, this theory would indicate:

- the sensory processing associated with individual sensors,
- how data from across a single sensor (say a vision sensor) is to be integrated
- how data from across a set of sensors of the same type (say two vision sensors) is to be integrated,
- how data from different sensor types (say vision and tactile sensors) is to be integrated.

In short, if one defines, for each sensor, an associated sensory information processing mechanism, forming what one can term “sensor modules,” then this theory will indicate not only the structure of each module (that is, its algorithms), but also the links that are to be forged when a number of these modules are assembled together to form a complex sensory system. Vision research to date has focused primarily on sensory information processing associated with single sensors, or multiple sensors of the same type. In this paper we wish to focus on the integration of sensory information drawn from sensors of different types.

In the immediately following section, the components of a multi-sensor machine are outlined. In section 3 a model for the sensory subsystems of these machines is outlined, and the goals of a Sensory Systems Theory are defined. In section 4 various forms of sensory integration are introduced and discussed with reference to the human sensory system. Finally, in section 5, research problems which need addressing are presented.

2. SENSORY SUBSYSTEMS

Assume an intelligent machine possessing a non-empty set of heterogeneous sensors. This machine is to be applied to solve a range of tasks. For each of these tasks an algorithm is developed and a corresponding computer program implemented. Each program embodies a particular behavioural pattern, so the set of programs together embodies the set of behaviours of the intelligent machine. For example, if there are ten tasks there will be ten programs and, therefore, ten behaviours. In the succeeding discussion we will refer to these programs as “behaviours” or “behavioural programs”.

Assume that each of these programs is independent of every other; in the sense that they do not share subroutines. Assume also that each program draws on a non-empty subset of the set of sensors possessed by the machine. In addition, assume that there is no preprocessing of sensory data prior to its access by each program. This means that each program directly accesses raw sensory data and embodies all the necessary signal processing required for it to make use of this raw data. Similarly for effectors with respect to control. This means that each program acts independently of the others, from the sensors through to the effectors (Fig. 1).

We will investigate this architecture now. What we will find are sensory processing requirements, at the sensory integration level, which could be provided as a central resource for these programs. What will emerge is a sensory integration database which can be accessed by these programs. This database will perform sensory integration continually in response to changes in the sensory signals. As such it will be continually updated. It is appropriate,

therefore, to refer to it as a “sensory integration engine”. Sensory Systems Theory can then be seen as the solution to the problem of designing this engine, for it will tell us how this engine is to be put together.

The process of extricating this sensory integration engine from the architecture developed above is as follows. We note first of all that extracting useful information from a particular sensor requires processing algorithms tailored to that sensor. Visual sensory signals, for example, undergo their own unique processing in the brain, as also do auditory and tactile sensory signals. We will assume that the precise form of this processing will be determined by the information sought from the sensor. It will also be assumed that a particular sensor can supply a number of items of information. The number will vary from sensor to sensor. We will also assume that for each item of information, one can define an algorithm for extracting that information from the sensor. Therefore, for each sensor one can define a set of algorithms for extracting useful information from that sensor.

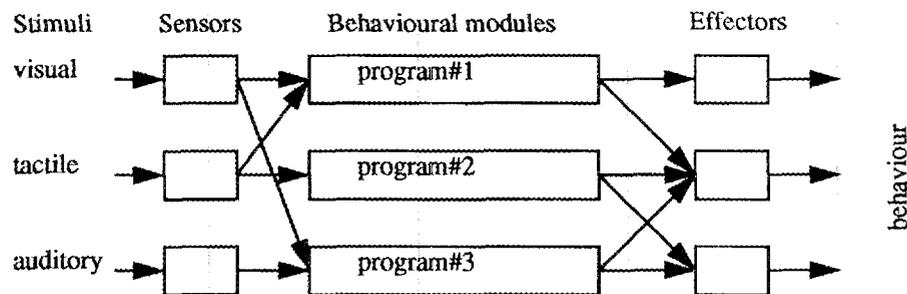


Fig. 1. Independent behavioural modules

As indicated above, each behavioural program will access a subset of the system’s sensors. Each sensor, therefore, will be a member of a number of these subsets. This means that a particular item of information may be requested from a particular sensor by a number of behavioural programs. If these programs embody all their own processing requirements, the algorithm for extracting an item of information from a particular sensor will be implemented a number of times. Similarly for other items of information associated with that sensor, and for other sensors. It makes sense, therefore, to provide this item of information as a central resource, and to decouple the extraction of useful sensory information from the individual behavioural programs which require that information. This will eliminate redundancy and improve efficiency.

This central resource will be a database. However, the information it provides will be extracted from the raw sensory data. This extraction process will be an integral component of the resource. Therefore, it is appropriate to refer to this resource as a “database engine”, though the sense in which “engine” is used here is different to its conventional use in the database community. A better term is “sensory integration engine”, but we will delay discussion of sensory integration until the following sections.

This engine may operate in one of two basic modes. In the first, it extracts an item of information from sensory data only when a behavioural program makes a request for that item of information. If the computing resources are available, a more efficient mode is to continually update that item of information as the sensory signal changes. It can then be made available immediately on request. If many items of information need to be extracted, this immediate response can only be achieved through some form of parallel architecture. For

example, each processing element of a parallel processor architecture may be dedicated to extraction of a single item of information.

We can usefully view this engine in terms of a conventional database. First, it will be queried by behavioural programs for information as and when that information is required by the program. Second, it will facilitate the development of further behavioural programs, a process which can be likened to database application software development. Third, developing this engine for a particular intelligent system, and therefore for a particular set of sensors, can be likened to the process of data analysis and modelling familiar in conventional database development.

A Sensory Systems Theory will define the structure of the sensory integration engine (database) just introduced. It will tell us, that is, how that engine is put together. Since we have associated a set of algorithms with each sensor, for the extraction of items of information from the corresponding sensory data, this integration amounts to little more than setting a set of processing modules (one for each sensor) side by side. "Integration" is the only required at the interface between the sensor modules and the behavioural programs, and this integration is trivial.

Behavioural programs may need to call on information, however, which is not directly available from individual sensors alone, but which can be derived from the integration of sensory data from a number of sensors. The form of this integration will be discussed shortly, but typical elementary examples include the extraction of depth information from a pair of visual sensors, the extraction of the direction of a sound from a pair of auditory sensors, and the determination of the direction of a sound measured with respect to visual space.

As before, a dedicated central resource could be made available for extracting this information. Our conception of the sensory integration engine presented above would correspondingly be adapted to accommodate this additional component. Now it is appropriate to refer to this central resource as a "sensory integration engine". In the modified conception, we retain the dynamic database idea, but it no longer consists of independent modules sitting side by side. Rather, links are forged between the modules to form an integrated structure (Fig. 2). Sensory Systems Theory will tell us where these links are, and how they are to be forged. It will tell us, that is, how to design a subsystem for sensory integration.

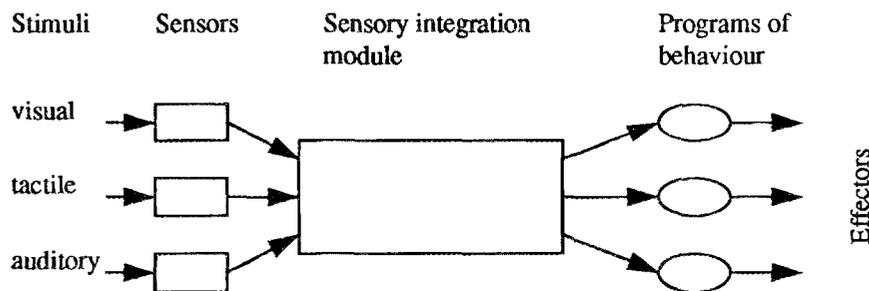


Fig. 2. Integrating sensory systems

3. THREE LEVELS OF INTEGRATION

From the previous section we have identified two levels of sensory information processing: the single-sensor level and the multi-sensor level. At each of these levels there is some form of sensory integration. At the single-sensor level, this integration involves gathering together data from different regions of the sensor (different regions of visual space for example). At the multi-sensor level it involves integration of data from different sensors, the classic example again being stereoscopic vision. In the following section these forms of integration will be discussed in more detail. In the present section we will introduce a third level of integration, which we will call “multi-modality” integration.

The term “multi-sensor” currently refers equally to a pair of visual sensors as to a combination of a visual sensor and an auditory sensor. However, there is an obvious distinction between these two cases. While a visual feature will stimulate both visual sensors, assuming the sensors have overlapping sensitivities and the stimulus falls within this region of overlap [1], a visual feature, or indeed an auditory feature, will not simultaneously stimulate both a visual sensor and an auditory sensor. This is due to the visual and auditory sensors being sensitive to different types of stimuli; electromagnetic radiation and sound, respectively. This in turn significantly alters the form of integration possible between two visual sensors and between a visual sensor and an auditory sensor.

To reflect this distinction we in turn distinguish between “multi-sensor” integration and “multi-modality” integration. Both involve multi-sensor integration. However, the first refers to sensors responding to the same stimulus type, for example two visual sensors, whereas the second refers to sensors responding to different stimulus types, for example a visual and an auditory sensor. The term “modality” is drawn from Physiology where it is used to refer to the different human sensing systems. Indeed, human beings are multi-modality sensing systems.

It is useful to view the introduction of this third level of integration in the context of specifying the sensory subsystem of an intelligent machine. Specifically, defining this sensory subsystem would involve specifying the sensory modalities possessed by the machine and then the components of each sensory modality. The latter in turn would include reference to the number of sensors possessed by each modality.

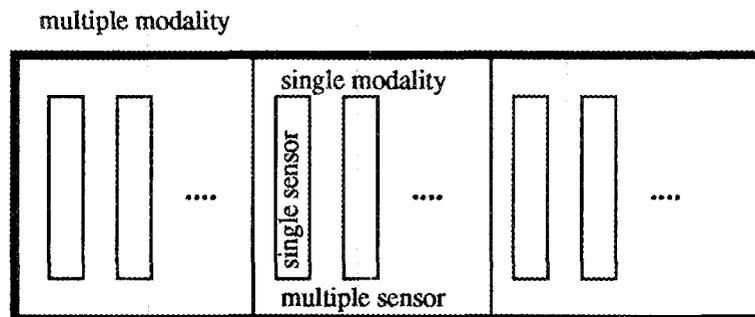


Fig. 3. Three levels of integration

It is apparent now that there are two important aspects to Sensory Systems Theory (Fig. 3). The first is the single modality aspect, and we see the objective of such a theory being that of developing a model for sensory information processing which is not tied to any one sensory modality, but says something about the processing of sensory information in all modalities.

This theory will accommodate sensory information processing within single sensors and across multiple sensors. The second is the multiple modality component, and we see the objective of such a theory being that of describing the mechanisms by which links can be forged between a set of sensory modalities to form a unified functional sensing module for an intelligent machine. In turn, three levels of sensor integration (single-sensor, single-modality multi-sensor, and multi-modality) need to be tackled by the theory. In the following section the forms of integration seen at each of these levels will be discussed in more detail.

4. TYPES OF INTEGRATION

4.1 SINGLE-SENSOR INTEGRATION

At the single-sensor level we will distinguish two basic forms of integration (Fig. 4). The first we will term “lateral integration”, and concerns integration of data from different “regions” of the same sensor. These regions may correspond, for example, to different regions of the electromagnetic spectrum (in the case of vision) or to different regions of the sensory space (different regions of a one- or two-dimensional visual image). This integration can operate on raw sensory data or on the products of processed sensory data, such as edge features. Familiar examples of integration at this level include region growing and edge chaining algorithms [2]. At the highest levels it involves integrating data about individual objects and sub-scenes in the context of forming an understanding of the complete scene captured by a sensor. The basic character of this form of integration, following from the examples above, is that of aggregation, or association, to contrast it with “sensor fusion”, which we will discuss presently.

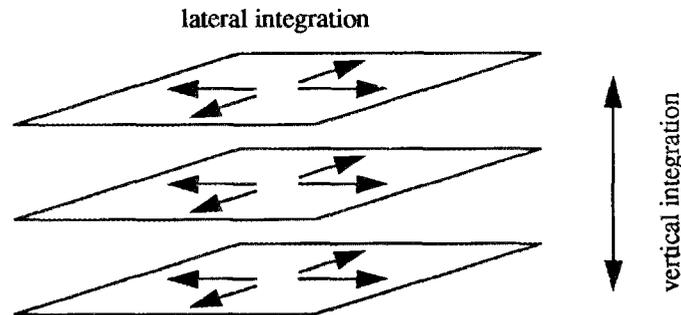


Fig. 4. Single-sensor integration

The second form of integration we will term “vertical” integration. It corresponds to the integration of shading, texture, motion and contour visual modules in the interests of extracting intrinsic images [3]. The stereo visual module is not included here for it is based on multiple sensors and is categorised, therefore, under multi-sensor integration. The basic character of this form of integration is that of “fusion”, in the sense, at least, that visual components are combined to form a single component which transcends the former, and there may be mutual modification of each of the former in order to achieve the latter. In the sense of combining a number of images to form a single image, it is like stereo fusion, but it does not depend on multiple sensors.

These two forms of integration, lateral and vertical, fit together in the following way. The lateral form of integration provides the mechanism by which the shading, texture, motion and contour visual modules are extracted from the raw visual image, making them available for vertical integration.

4.2 MULTI-SENSOR INTEGRATION

One of the characteristic features of single-sensor integration is that multiple images are produced from the same raw visual image. In consequence, all of these generated images are in “register” with the original image and with each other. In moving from the single-sensor level to the multi-sensor level, and indeed to the multi-modality level, this registration is no longer given, and indeed much of the problem of integration at these levels concerns bringing images, obtained from different sensors, into registration.

The product of this integration is generally further useful information. The two cases in point are vision and audition. In stereoscopic vision data from two visual sensors is combined to give information about the three-dimensional structure of visual space (Fig. 5). This generates the stereo visual module which was mentioned above. In audition, on the other hand, integration of data from two auditory sensors enables information about the direction of sounds in space to be generated.

Example: vision

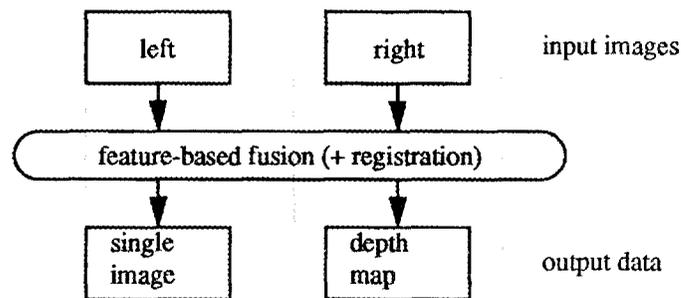


Fig. 5. Multi-sensor integration

While registration is what is to be achieved, the mechanism for achieving that registration depends on there being a representation of the same feature(s) in the two images simultaneously. In stereoscopic vision the features are usually taken to be lines and edges. Finally, the basic character of multi-sensor integration is that of a generator of new, useful, information. At the same time, though, there is “fusion”, since two separate images combine to form a single image. Whatever its useful product may be, however, multi-sensor fusion is at least registration.

4.3 MULTI-MODALITY INTEGRATION

Two forms of integration can be identified at the multi-modality level (Fig. 6). The first is “spatial integration”. This involves bringing the spaces of the different sensory systems into register with each other. It is equivalent to the registration seen at the multi-sensor level, but is between sensors sensitive to different types of sensory stimuli. The exemplar is visual-

auditory integration. It is characterised, generally, by the lack of a feature-based mechanism for achieving registration. That is, although the same object may stimulate the visual and auditory sensors simultaneously (for example a person speaking), the same feature will not be represented in both senses. Also, multi-modality integration contrasts with multi-sensor registration in that the registration is more of a “mapping” than an “alignment” of sensory fields. There are circumstances, however, where this form of multi-modality integration reduces to multi-sensor integration.

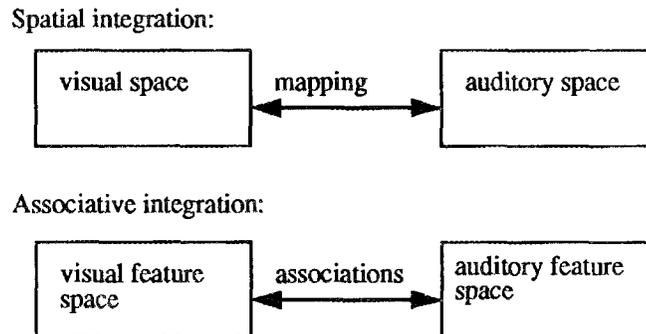


Fig. 6. Multi-modality integration

An example is integrating visual and range sensory systems. A ranging sensor produces a two-dimensional depth map. At the raw image level there is no basis for integrating these sensory systems, for they do not respond to the same features. This justifies regarding them as distinct modalities. However, luminance discontinuities in the visual image may correspond to depth discontinuities in the range image. If these discontinuities are extracted from each image separately, they can then be used to achieve registration, and therefore visual space can be mapped into range space, and vice versa.

The second form of multi-modality integration we term “associative” integration. Associative integration mediates the high-level transformation of features from one sensory modality to another, and vice versa. The typical example is hearing a voice and associating it with a face which is not currently in the visual field, or vice versa. Another example is being able to picture in one’s mind the visual form of an object which is manipulated out of sight. The term “associative” is used to describe this form of integration because what binds the features belonging to the different sensory modalities together is their co-occurrence with each other in the environment (for example, a face associated with a voice).

There is significantly more to associative integration than this example would indicate.. Picturing in the mind’s eye an object perceived through tactile manipulation may require piecing together individual tactile-visual associations to create a picture of an object which, although encountered tactually, may not previously have been encountered visually. In other words, the transformation may not necessarily depend on a prior association of features belonging to separate modalities, but should be seen, rather, as a problem solving process.

Also, the association may in certain cases be complex, involving not two sensory modalities, but a number of sensory modalities. In the example just described, the proprioceptive sensory modalities are also required, for they provide information about the posture of the hand and arm manipulating the object. If a three-dimensional visual representation of the object is to be

achieved, this proprioceptive information needs to be available, needs to be integrated with the tactile sensory modality, and the two used to assemble a visual representation of the object.

In visual-auditory modality integration, another form of transformation across modalities is also found: the transformation of a symbolic description of an object into a visual representation. This symbolic description may be obtained through the spoken word (the auditory sensory modality) or the written symbol (the visual sensory modality). Obviously the construction of a visual representation from the latter would not seem to involve a modality other than the visual modality. On the other hand, the written symbols may first be transformed into their auditory counterparts, and these latter used as the basis for picturing the object captured by the description, in which case the visual and auditory modalities are involved in a complex multi-modality integration process (Fig. 7). In general, however, the basic character of multi-modality integration is one of mapping and association.

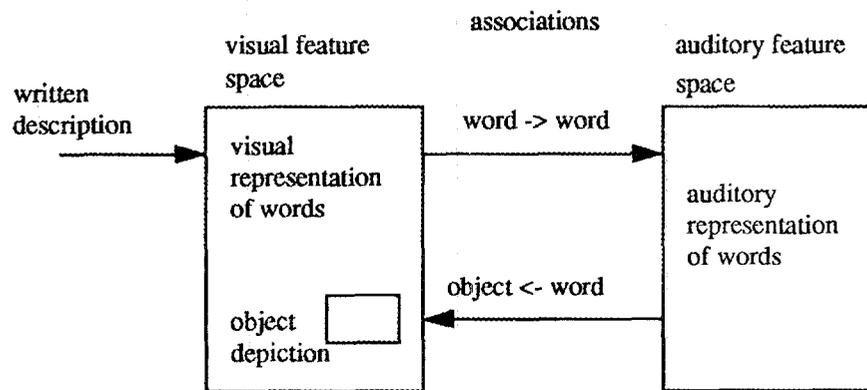


Fig. 7. Visualisation via associative integration

Table 1 summarises the types of integration discussed in this section.

TABLE 1. Types of Integration

<u>level of integration</u>	<u>type of integration</u>
single sensor	lateral vertical
multiple sensor	registration (feature-based) (non feature-based)
multiple modality	spatial associative

5. DISCUSSION

In summary, we have proposed a model for the sensory subsystems of intelligent machines, consisting of three levels of integration: single sensor, multiple sensor, and multiple modality. We define the objective of a Sensory Systems Theory as that of developing a theoretical understanding of the integration of sensory information within single sensors, across multiple

sensors of a single modality, and across multiple modalities. The theory should tell us, first, the structure of a single sensory modality and, second, how to forge links between a number of sensory modalities to create a multiple modality sensory subsystem for an intelligent machine.

It should be noted that the sensory subsystem model we have proposed is not a procedural model. We are not partitioning the sensory subsystem into a single-sensor processing module, providing inputs to a multi-sensor processing module and this in turn driving a multi-modality processing module. In many instances this processing model will make sense, but the whole objective of developing a Sensory Systems Theory is to tell us what the appropriate processing model for a particular sensory subsystem is. Thus, although we separate the stereo visual module from the shading and texture visual modules, associating the former with the multi-sensor level and the latter with the single-sensor level, we by no means imply that the former is in a processing module further down the line from the latter. On the contrary, indeed, the extraction of the stereo module begins immediately the visual information enters the visual cortex of the brain [4].

The basic objectives of a Sensory Systems Theory include developing a generic model for the integration of data within a single sensory modality. This at first seems an enormous task given the diversity in sensors found both at the single modality level and across multiple modalities. On the one hand, for example, the theory would have to cope with a single modality where the sensors are the same (for example, two human-like visual sensors) or different (for example, a human-like visual sensor and an infra-red sensor), and with any number of each. On the other hand, it would have to cope with sensory modalities as diverse as the visual, auditory and tactile modalities. Nevertheless, the advantages to be gained are enormous.

One advantage would be the ability to develop generic computer architectures and algorithms for sensory information processing which could be deployed flexibly in the development of modality-specific information processing mechanisms, and could accommodate a number of different sensory modalities within the same architecture. An extension to this architecture to take account of multiple modality integration would then enable it to be used for the complete sensory subsystem of an intelligent machine.

The greatest strain on achieving multi-modality integration within such a generic architecture will be "spatial" integration; mapping together the "sensory spaces" of different sensory modalities. In the case of integrating visual and auditory space this may reduce to simple geometry. In the case of visual and tactile, however, the form of spatial integration required may be much more sophisticated. Research, therefore, is required to determine the precise form of this visual-tactile integration, and indeed to determine the nature of other forms of spatial integration.

The issue of spatial mapping poses a major research problem, but motivates also a distinction to be drawn in Sensory Systems Theory between the mechanisms of sensory integration and the development of those mechanisms. The question raised by the latter is how, in humans or other animals, the integration of the sensory systems has developed. Multi-modality spatial integration is particularly interesting since there is no apparent basis for it. For example, the auditory sensory modality does not respond to visual stimulation, and the visual sensory modality, in turn, does not respond to auditory stimulation; nevertheless, in human beings the two sensory spaces are mapped together, such that visual gaze can be oriented to fixate on the source of an auditory stimulus.

If there is no basis in the features stimulating the two sensory modalities for integration, we must look elsewhere. An obvious place to look, though it may not be the only possible solution to the problem, is evolution. The assumption is that evolution favoured the development of a visual-auditory spatial mapping. If so, there is an obvious implication for the engineering of intelligent machines, for it means that these spatial mappings will need to be preprogrammed prior to the introduction of the machine into its target environment.

A counter to this might be the mapping together of a visual and a range sensory modality, where edges of different types (luminance and depth) can form the basis for integration, and the same mechanism may operate for integration in this case as for integration across two visual sensors in stereoscopic vision. That is, simple feature-based registration. But this in turn raises the issue of the mechanism by which this form of integration develops. That is, how do corresponding edges in images from different sensors know that they are one and the same.

This distinction is also motivated by the converse requirements of robustness. If a mapping has been established, and some event then intervenes to disrupt the mapping, it is observed in humans [5], and it is desirable in machines, that the situation be retrieved. This calls for an adaptive ability on the part of the machine. It may happen that this adaptive ability is the very same that supports the development of the mapping in the first place. Research is required to resolve these issues.

A useful paradigm for robustness here is eye-hand coordination. This depends on mapping the space of arm postures onto visual space. It would appear that the basis for this is partly innate and partly experiential. In essence, innateness provides a crude mapping which experience fine tunes. An event which might intervene to disrupt this mapping, for example attaching it to the hip rather than the shoulder, would give the arm a new placement relative to the visual sensors, requiring a completely new mapping. Retrieving this disruption would require an adaptive mechanism which might need facilities above and beyond those for the experiential fine tuning of the crude innate mapping.

An understanding of both the mechanism of integration and the processes by which those mechanisms develop is particularly important in the context of engineering intelligent machines. Its importance lies in establishing a trade-off between the "pre-programming" of the sensory subsystem prior to its introduction into the environment in which it will operate, and the subsequent "experiential learning" required to bring it to its desired level of performance. From a programming point of view, the objective would be to reduce the amount of prior programming and, therefore, to leave as much of the development of sensory integration mechanisms to experience. This in turn puts a major emphasis on developing a suitable substrate in which this experiential learning can take place.

Taking into account the requirement for robustness, and assuming that this robustness depends on mechanisms other than those required for the development of sensory integration mechanisms, three aspects to the problem of developing sensory integration mechanisms for intelligent machines can be identified: preprogramming, development (training), adaptability (robustness). To solve these problems a good starting point would be to study human sensory integration. Here the corresponding issue to the relation between preprogramming and experiential learning, is the relation between nature and nurture. How much of human behaviour, that is, is due to innate mechanisms and how much is due to experience?

Consider visual-auditory integration as an example. When a child hears a click to the right or left of its head it orients in such a way as to direct its gaze in the direction of the sound. This is

an example of the mapping of the auditory and visual spaces together, so that a signal in one can be located in the space of the other. This orienting behaviour is an innate response mediated by the Superior Colliculi in the Thalamus of the brain. Neurophysiological experiments clearly show the mapping together of visual and auditory stimuli in the Superior Colliculi [6].

This innate mechanism mediates the development of higher level mechanisms of visual-auditory spatial integration in the cerebral cortex. In turn, it provides a fall-back mechanism when the former fails. We can see here, therefore, a trade-off between innate preprogramming and experiential learning. Whether this mechanism represents the optimal trade-off between innateness and experience is another matter, though one would favour an affirmative answer given the remarkable achievements of evolution in other respects.

This innate versus experiential trade-off can be seen in other facets of development, including eye-hand coordination mentioned above. A first step to understanding this trade-off and the mechanisms for achieving both innate preprogramming and experiential learning, would obviously be, therefore, a study of the corresponding human mechanisms. This is a useful starting point, therefore, for pursuing a Sensory Systems Theory.

Finally, in this paper we have focused on the sensory subsystem of a multi-sensor machine. The theory suggested here will be just one component of a more elaborate intelligent machine theory. This more elaborate theory will include details of the representation of knowledge and skill, the integration of sensory data with effector actions, and ultimately the coordination of sensors and effectors under the guidance of knowledge and skill. It will tell us how to put together intelligent machine. As part of that theory, the Sensory Systems Theory will tell us how to put together one vital component of that machine.

REFERENCES

- [1] G. McKee. What can be fused? In J. K. Aggarwal, editor, *Multisensor Fusion for Computer Vision*. Springer Verlag. To be published.
- [2] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, 1982.
- [3] J. Y. Aloimonos and D. Shulman. *Integration of Visual Modules: An Extension of the Marr Paradigm*. Academic Press, 1989.
- [4] G. F. Poggio and B. Fischer. Binocular interaction and depth sensitivity in striate and prestriate cortex of behaving rhesus monkey. *Journal of Neurophysiology*, 40(6): 1392-1405, 1977.
- [5] R. Held. Plasticity in sensory-motor systems. *Scientific American*, November 1965.
- [6] M. A. Meredith and B. E. Stein. Interactions among converging sensory inputs in the superior colliculus. *Science*, 221:389-391, July 1983.

DATA STRUCTURES + GENETIC OPERATORS = EVOLUTION PROGRAMS

Zbigniew Michalewicz, Joseph Schell, and David Seniv

Department of Computer Science
University of North Carolina
Charlotte, NC 28223, USA

ABSTRACT

This paper discusses a new approach for solving constrained optimization problems using nonstandard genetic algorithms. We call this approach: “evolution programming”. We argue that a “natural” representation of a solution for a given problem plus a family of applicable genetic operators might be quite useful in the approximation of solutions of constrained optimization problems. We describe some experiments of using this approach.

1 INTRODUCTION

To solve a nontrivial problem using a genetic algorithm approach we can either transform the problem (it need not be an easy task) into a form appropriate for the genetic algorithm, or we can transform the genetic algorithm to suit the problem. This paper represents the latter approach. We discuss applications of nonstandard genetic algorithms to approximate constrained optimization problems. We depart from classical genetic algorithms which operate on strings of bits: rather, we search for richer data structures and applicable “genetic” operators for these structures.

The binary alphabet offers the maximum number of schemata per bit of information of any coding (see [5]) and consequently the bit string representation of solutions has dominated genetic algorithm research. This coding also facilitates theoretical analysis and allows elegant genetic operators. But the ‘implicit parallelism’ result does not depend on using bit strings (see [1]) and it may be worthwhile to experiment with richer data structures and other types of genetic operators. We argue here that these modifications may be useful when the problem to be solved involves non-trivial constraints that continually have to be maintained during the genetic operations.

We believe that a promising direction for incorporating constraints for genetic algorithms is with the introduction of richer data structures together with a family of applicable “genetic” operators, which would “hide” the constraints present in the problem. These richer data structures, with appropriate genetic operators, would constitute an *evolution*

program. The structure of an evolution program is identical to the structure of a classical genetic algorithm. The differences are hidden on the lower level: each chromosome need not be represented by a bit-string. Moreover, for the recombination process we introduce “genetic” operators appropriate for the given structure and the given problem.

Three experiments (the transportation problem, the graph drawing problem, and the traveling salesman problem) based on the proposed methodology are discussed in the paper. All of these adopt “natural” data structures and specialized “genetic” operators. The results are more than encouraging. We discuss them in turn.

2 EVOLUTION PROGRAM FOR THE TRANSPORTATION PROBLEM

In this section we describe an optimization problem, known as the transportation problem, and show how it can be formulated as an evolution program.

2.1 THE TRANSPORTATION PROBLEM

Suppose that a commodity is available at a number of sources and that certain quantities of this commodity are required at a number of destinations. The demand at each destination may be satisfied from one or more sources. The objective of the transportation problem is to determine the amount to be shipped from each source to each destination such that the total transportation cost is minimized.

If the transportation cost on every route is directly proportional to the number of units transported, we have a *linear transportation problem*. Otherwise, we have a *nonlinear transportation problem*.

Suppose that there are n sources and k destinations, that the amount of supply at source i is $sour[i]$ and the demand at destination j is $dest[j]$, and that the unit transportation cost between source i and destination j is given as a function f_{ij} .

Let x_{ij} be the amount transported from source i to destination j ; then the transportation problem is to minimize

$$\sum_{i=1}^n \sum_{j=1}^k f_{ij}(x_{ij}),$$

subject to the following constraints:

- (1) $\sum_{j=1}^k x_{ij} = sour[i]$, for $i = 1, \dots, n$
- (2) $\sum_{i=1}^n x_{ij} = dest[j]$, for $j = 1, \dots, k$
- (3) $x_{ij} \geq 0$, for $i = 1, \dots, n$ and $j = 1, \dots, k$

This, in fact, is the *balanced* transportation problem, due to equalities in (1) and (2). If all the $sour[i]$'s and $dest[j]$'s are integer, any optimal solution to a balanced linear transportation problem is an integer solution, *i.e.* all x_{ij} ($1 \leq i \leq n$, $1 \leq j \leq k$) are integers. Moreover, it can be shown that the number of non-zero values among the x_{ij} 's is at most $k + n - 1$. However, it is not the case for a nonlinear transportation problem, where x_{ij} 's need not be integers and the number of non-zero elements may be arbitrary.

2.2 FORMULATING THE TRANSPORTATION PROBLEM AS AN EVOLUTION PROGRAM

In order to build an evolution program for the transportation problem, we need to find a representation for candidate solutions and create appropriate genetic operators for this representation.

It seems that for the transportation problem a matrix representation is clearly the most natural one — after all, this is how it is presented and solved by hand. So let us assume a matrix $V = (v_{ij})$ ($1 \leq i \leq k, 1 \leq j \leq n$) represents a possible solution to the transportation problem.

There is a large group of possible “genetic” operators we can apply to matrices. Different operators may be selected for linear and nonlinear cases of the transportation problem. Let us consider these two cases separately.

2.2.1 Linear Transportation Problem

We search for a solution expressed as a table of nonnegative integers. Because of nontrivial constraints, we can create the following “genetic operators”:

- **mutation:** this operator would select part of a matrix, find marginal sums, erase all entries in the selected part, and place some random integers for all entries such that the new numbers satisfy constraints for marginal sums.
- **arithmetical-crossover:** this operator would create a matrix which is an arithmetical average of two parent matrices. Additionally, the resulting matrix (which need not contain only integers) is rounded in a special way to preserve all marginal constraints.

We have built an evolution program for solving the linear transportation problem using a matrix structure and the above operators [17]. A number of examples from the transportation problem chapters of textbooks in Operations Research were chosen as the base set of problems. They were supplemented by a number of other examples with randomly generated unit costs, supply values, and demands. For the purpose of evaluation of the evolution program, each example was first solved using a standard transportation algorithm so that the optimum value was known for later comparison. Since the optimum transportation plan in the linear case can be determined easily, we have selected the percent above optimum in 100 generations as an evaluation of the “goodness” of our approach. In all cases, this number was below 2%.

For a further discussion on the linear transportation problem and possibilities of applying classical genetic algorithms to this problem, see [17].

2.2.2 Nonlinear Transportation Problems

We have investigated the effectiveness of our approach dealing with nonlinear transportation problems. This leads to further opportunities in selecting genetic operators. We have created the following “genetic operators”:

- **mutation-1:** this operator would select part of a matrix, find mariginal sums, erase all entries in the selected part, and place some random integers for all entries such that the new numbers satisfy constraints for mariginal sums. At the same time, this operator attempts to introduce as many zero entires into the matrix as possible.
- **mutation-2:** this operator is identical to the previous one except it avoids choosing zero entries by selecting values from a range.
- **arithmetical crossover:** this operator is simpler than its counterpart for integer numbers. Two matrices V_1 and V_2 would produce two offspring, W_1 and W_2 , such that $W_1 = c_1 \cdot V_1 + c_2 \cdot V_2$, and $W_2 = c_2 \cdot V_1 + c_1 \cdot V_2$, where c_1 and c_2 are any positive reals such that $c_1 + c_2 = 1$. Note that this operator would preserve the constraints (sums for rows and columns).

The experimental application of this approach for solving nonlinear transportation problems is more than encouraging. We compared the results obtained using a commercial system, GAMS (see [2]), with our evolution program (called GENETIC-2) on six nonlinear cases (nonlinear functions A – F). For a full discussion on the selection and classification of these functions, see [12].

A typical comparison of the optima between GENETIC-2 (averaged over 5 seeds) and GAMS for all six functions is shown in the table below.

Function	GAMS	GENETIC-2	% difference
A	281.0	202.0	–28.1%
B	180.8	163.0	–9.8%
C	4402.0	4556.2	+3.5%
D	408.4	391.1	–4.2%
E	145.1	79.2	–45.4%
F	1200.8	201.9	–83.2%

3 EVOLUTION PROGRAM FOR THE GRAPH DRAWING PROBLEM

In this section we describe the graph drawing problem and show how it can be formulated as an evolution program.

3.1 THE GRAPH DRAWING PROBLEM

The graph drawing problem (see [4]) is the determination of an algorithm for drawing pictorial diagrams of a directed graph which is easy to understand and remember. A large number of algorithms have been proposed for drawing graphs. The kinds of algorithms used, and their costs, vary according to the class of graph for which they are intended (*e.g.* trees, planar graphs, hierarchical graphs or general undirected graphs), the aesthetic criteria they consider, and the methods they use for optimizing the layout. In most cases, finding optimal layouts for large graphs is prohibitively expensive, so a number of heuristic methods have been investigated that find approximate solutions in a reasonable amount of time. A good discussion of the problem of drawing graphs, aesthetic criteria that have been considered, and various methods that have been proposed is given in [16]. A more extensive bibliography is given in [4].

The aesthetic criteria (for ease in understanding and remembering) can be viewed as goals of the optimization problem and include:

- C_1 : Arcs pointing upward should be avoided,
- C_2 : Nodes should be distributed evenly over the page,
- C_3 : There should be as few arc crossings as possible.

3.2 FORMULATING THE GRAPH DRAWING PROBLEM AS AN EVOLUTION PROGRAM

The representation of a solution for the graph drawing problem consists of a $2 \times N$ matrix which stores the row and column coordinates of each node on a page (N is the total number of nodes). Figure 1(a) give an example of 18 nodes graph and Figure 2 provides its genetic representation.

The evaluation of each chromosome M is based on three aesthetic criteria discussed earlier and is expressed as:

$$Eval(M) = a_d \cdot n_d(M) - a_c \cdot n_c(M) + a_l \cdot n_l(M)$$

where a_d , a_c , and a_l represent the weights associated with arcs pointing down, arcs crossing, and nodes that lie on the same level; $n_d(M)$ and $n_c(M)$ denote the numbers of such arcs; $n_l(M)$ denote the number of nodes that lie on the same level.. The horizontal arcs are handled the same as arcs that point upward.

The “genetic” operators used in the system were:

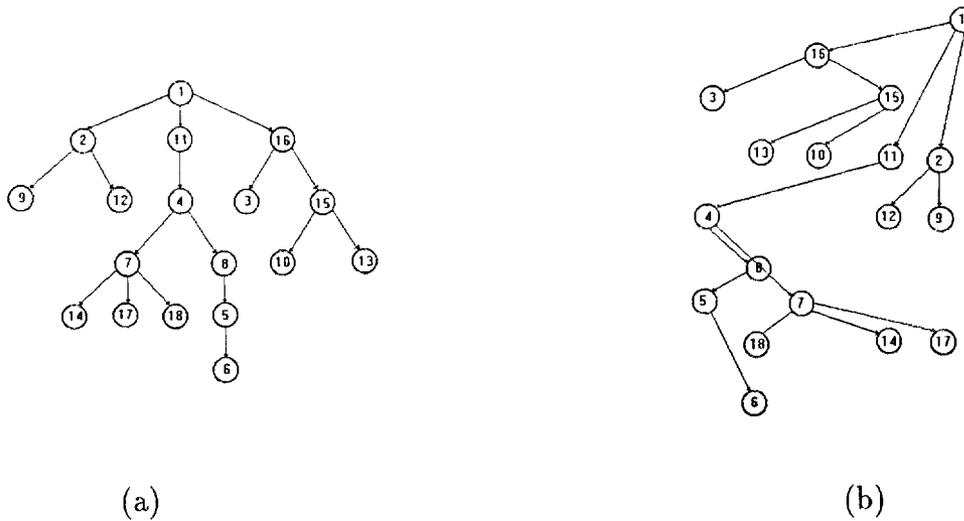


Figure 1: Diagram of graph G (a); output from evolution program for the same graph (b).

node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
row	1	2	3	3	5	6	4	4	3	4	2	3	4	5	3	2	5	5
col	5	2	7	5	6	6	4	6	1	8	5	3	10	3	9	8	4	5

Figure 2: Genetic representation of diagram of graph G .

- **standard mutation:** this operator changes randomly a node's coordinate: either row or column.
- **smart mutation:** this operator attempts to use problem specific knowledge to mutate a chromosome. It focuses on getting all arcs pointing down by moving nodes without parents up, and positioning child nodes below their parents.
- **crossover:** this operator takes a random number of nodes from the first parent and remaining nodes from the other parent. If the row and column of a node from the second parent is already represented in nodes from the first parent, then the node is randomly assigned.

The results are quite interesting: Figure 1(b) provides the shape of the best chromosome after 200 generations for graph G . This graph has no arcs pointing up or horizontal, one arc that crosses another, and 11 of 13 siblings are on the correct level.

Two other "evolution programs" (one of these uses a $r \times c$ matrix for its chromosomes, where r and c are the number of rows and columns available on the output page; both

programs use different “genetic operators”) for the graph drawing problem are described in [8].

4 EVOLUTION PROGRAM FOR TRAVELING SALESPERSON PROBLEM

In this section we describe the traveling salesman problem and show how it can be formulated as an evolution program.

4.1 THE TRAVELING SALESMAN PROBLEM

The statement of the TSP is simple: a traveling salesman must visit every city in his territory exactly once and then return to the starting point; given the cost of travel between all pairs of cities, how should he plan his itinerary so that the total cost of his entire tour is minimum?

4.2 FORMULATING THE TRAVELING SALESMAN PROBLEM AS AN EVOLUTION PROGRAM

The representation of a solution for TSP is a two-dimensional binary matrix $V = (x_{ij})$. If the tour goes from the city i directly to the city j , then $x_{ij} = 1$, otherwise $x_{ij} = 0$. This means that there is only one nonzero entry for each row and each column in the matrix (for each city c there is exactly one city visited prior to c , and exactly one city visited next to c). For example, a chromosome in Figure 3(a) represents a tour that visits the cities (1, 2, 4, 3, 8, 6, 5, 7) in this order. Note also that this representation avoids the problem of specifying the starting city, *i.e.* the Figure 3(a) represents also tours (2, 4, 3, 8, 6, 5, 7, 1), (4, 3, 8, 6, 5, 7, 1, 2), *etc.*

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	1
4	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	1	0
6	0	0	0	0	1	0	0	0
7	1	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0

(a)

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0
7	1	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0

(b)

Figure 3: Binary Matrix Chromosomes

It is interesting to note, that each complete tour is represented as a binary matrix with only one bit in each row and one bit in each column set to one; however, not every matrix

with these properties would represent a single tour. Binary matrix chromosomes may represent multiple sub-tours: each sub-tour will eventually loop back onto itself, without connecting to any other sub-tour in the chromosome. For example, the chromosome from Figure 3(b) represents two sub-tours

(1, 2, 4, 5, 7) and (3, 8, 6).

We decided to allow sub-tours in the hope that natural clustering would take place. After the NGA algorithm had terminated, the best chromosome would be reduced to a single tour by successively combining pairs of sub-tours using a deterministic algorithm. Sub-tours of one city (a tour leaving a city to travel right back to itself) having a distance cost of zero would make no sense and were not allowed. We arbitrarily set a lower limit of three cities in a sub-tour, in an attempt to prevent the GA from reducing a TSP problem to a large number of sub-tours each with very few cities.

To demonstrate the significance of this representation, and of allowing sub-tours to exist within chromosomes, the example in Figure 4 was devised. Figure 4(a) depicts the sub-tours resulting from a sample run of the algorithm on a number of cities intentionally placed in clusters. As expected, the algorithm developed isolated sub-tours. Figure 4(b) depicts the tour after the sub-tours have been combined.

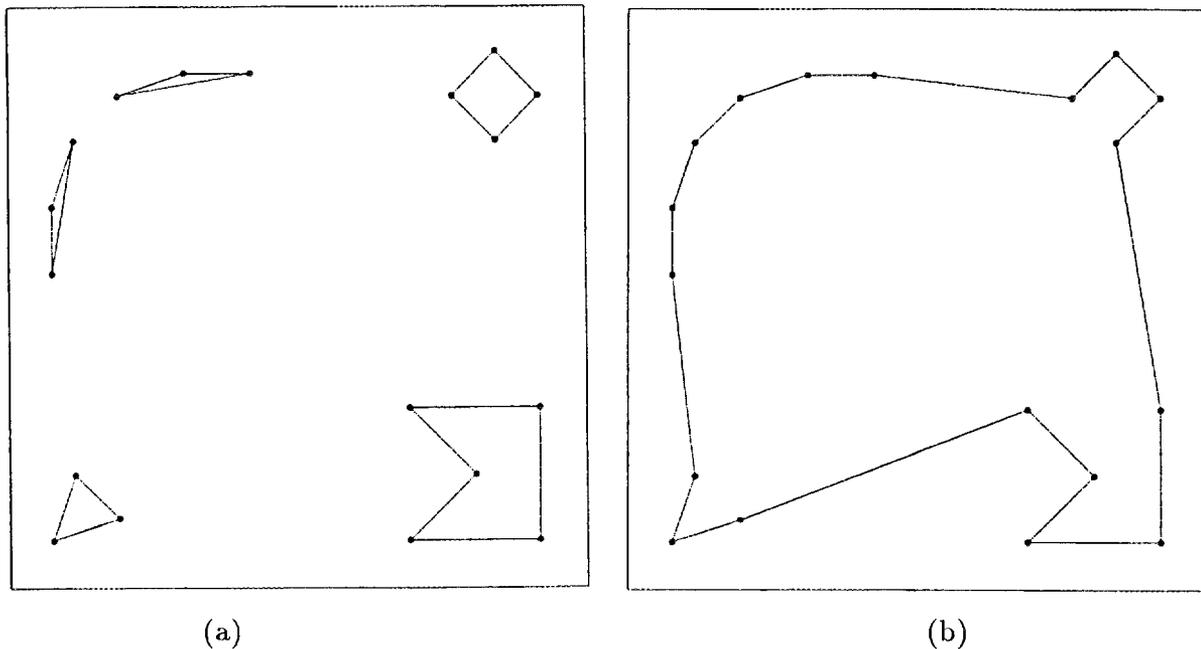


Figure 4: Separate sub-tours (a) and the final tour (b)

The “genetic” operators used in the system were:

- **mutation:** this operator takes a chromosome, randomly selects several rows and columns in that chromosome, removes the set bits in the intersections of those rows and columns, and randomly replaces them (in possibly a different configuration).
- **crossover:** the crossover operator begins with a child chromosome that has all bits reset (zero). The operator first examines the two parent chromosomes, and when it discovers the same bit (identical row and column) set in both parents, it sets a corresponding bit in the child. The operator then alternately copies one set bit from each parent, until no bits exist in either parent which may be copied without violating the basic restrictions of chromosome construction. Finally, if any rows in the child chromosome still do not contain a set bit, the chromosome will be filled in randomly. As the crossover traditionally produces two child chromosomes, the operator is executed a second time with the parent chromosomes transposed.

In an attempt to evaluate the results of our algorithm, we used an empirical formula for the expected length of L^* of a minimal TSP tour:

$$L^* = K\sqrt{N \cdot R},$$

where N is the number of cities, R is the area of the square box within which the cities were randomly placed, and K is an empirical constant of approximately 0.765. The square box to contain the random cities was selected to be 13.071895 units per side. This resulted in an L^* of 100.00.

Typical results from the algorithm, as applied to 100 cities randomly placed, are displayed in Figure 5(a), where the resultant chromosome contained 12 subtours, with a combined cost of 108.3. After the subtours were combined into a single tour, the cost of the entire tour was 112.9 (Figure 5(b)).

The early results are promising, since they are only slightly worse than those reported in [7], where 20,000 generations were used (twice as much). (For more details the reader is referred to [15]). Additionally, the proposed method leaves some room for further improvements. Firstly, our deterministic algorithm for combining several sub-tours into a single tour is far from perfect (see, for example, crossing lines on Figure 5(b): these can be easily removed rearranging the sequence of nodes to be visited). Secondly, there are other “genetic” operators on binary matrices, which may be even better than the current ones for the TSP. Currently, we explore this possibility further.

CONCLUSIONS

We plan to build evolution systems for different problems, using different structures and different operators. Later, all systems would be combined in a single software product suitable for various types of optimization. The only responsibility of a user (apart from supplying the evaluation function) would be to select an appropriate data structure and

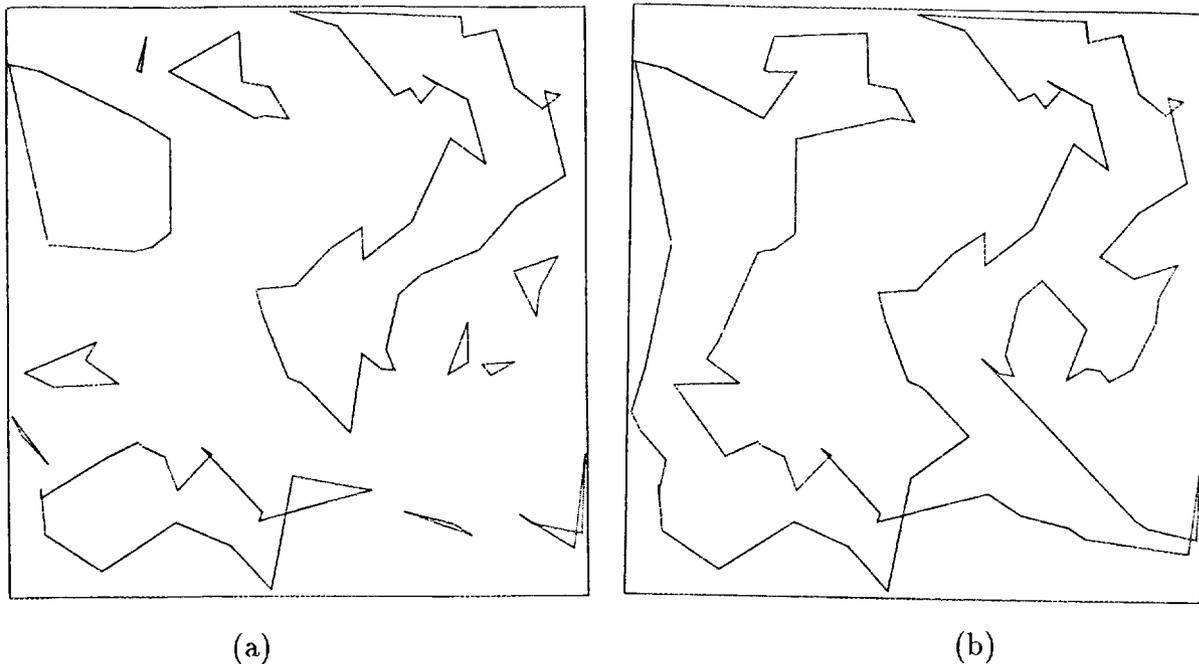


Figure 5: The best chromosome (a) and the final tour (b)

meaningful genetic operators, the latter selected from a library provided for each data structure.

It is too early to give convincing evidence of the soundness of the proposed approach; however, the first results are very encouraging. Additionally, it seems that that a “natural” representation of a solution for a given problem plus a family of applicable genetic operators might be more efficient in solving some constrained optimization problems

ACKNOWLEDGMENTS

This research was supported by a grant from the North Carolina Supercomputing Center.

References

- [1] Antonisse, J., *A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint*, in [14], pp.86–91.
- [2] Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User’s Guide*, The Scientific Press, 1988.
- [3] Davis, L., (Editor), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987.

- [4] Eades, P., and Xuemin, L., *How to Draw a Directed Graph*, Technical Report, Department of Computer Science, Brown University, 1988.
- [5] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.
- [6] Grefenstette, J.J., (Editor), *Proceedings of the First International Conference on Genetic Algorithms*, Pittsburg, July 24–26, Lawrence Erlbaum Associates, Publishers, 1985.
- [7] Grefenstette, J.J., *Incorporating Problem Specific Knowledge into Genetic Algorithms*, in [3], pp.42–60.
- [8] Groves, L., Michalewicz, Z., Elia, P., Janikow, C., *Genetic Algorithms for Drawing Directed Graphs*, *Proceedings of the Fifth International Symposium on Methodologies of Intelligent Systems*, Knoxville, pp.268–276, October 25–27, 1990.
- [9] Janikow, C., and Michalewicz, Z., *Specialized Genetic Algorithms for Numerical Optimization Problems*, *Proceedings of the International Conference on Tools for AI*, Washington, pp.798–804, November 6–9, 1990.
- [10] Michalewicz, Z. and Janikow, C., *Genetic Algorithms for Numerical Optimization*, *Statistics and Computing*, Vol.1, No.1, 1991.
- [11] Michalewicz, Z., Kazemi, M., Krawczyk, J., Janikow, C., *Genetic Algorithms and Optimal Control Problems*, *Proceedings of the 29th IEEE Conference on Decision and Control*, Honolulu, pp.1664–1666, December 5–7, 1990.
- [12] Michalewicz, Z., Vignaux, G.A., and M. Hobbs, *A Genetic Algorithm for the Non-linear Transportation Problem*, *ORSA Journal on Computing*, Vol.3, No.4, 1991.
- [13] Michalewicz, Z. and Janikow, C., *GENOCOP: Genetic Algorithm for Numerical Optimization of Constrained Problems*, to appear in the *Communications of the ACM*, 1991.
- [14] Schaffer, J. David (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, June 4–7, 1989, Morgan Kaufmann Publishers, 1989.
- [15] Seniv, D. and Michalewicz, Z., *A Non-Standard Genetic Algorithm for Traveling Salesman Problem*, UNCC Technical Report, 1991.
- [16] Tamassia, R., Di Battista, G. and Batini, C., *Automatic Graph Drawing and Readability of Diagrams*, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.18, No.1, pp.61–79.
- [17] Vignaux, G.A. and Michalewicz, Z., *A Genetic Algorithm for the Linear Transportation Problem*, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.21, No.2, 1991.

Generic Problem Solving Models in a Computer Aided Methodology for the Construction of Knowledge Based Systems

André R. Probst

University of Lausanne (HEC) and IBM Switzerland

Alex I. Horvitz

University of Lausanne (HEC) and Banque Cantonale Vaudoise, Lausanne, Switzerland

Dieter Wenger

Swiss Bank Corporation, Basel, Switzerland

ABSTRACT

This paper describes a Computer Aided Knowledge Engineering (CAKE) tool that embodies a development methodology specially suited for the construction of knowledge based systems. The computer based tool also includes a library of predefined problem solving models called G-TECS (Generic Techniques). Categories of G-TECS are defined as classes and are organized in hierarchies. The inheritance and instantiation mechanism substantially ease the generation of specific applications. Reusability of predefined models is supported at a broad level.

1. INTRODUCTION

THE NEED FOR COMPUTER AIDED KNOWLEDGE ENGINEERING TOOLS

Knowledge-based systems (KBSs) are, despite the myths and hypes which accompanied their early stages, information processing systems (IPS). It is true that there are significant differences between implementation techniques used for constructing KBSs and traditional IPSs. However, the great deal of experience gained in the field of information system development is still valid and very useful.

Developers of knowledge-based systems must profit from the years of experience in developing IPSs. Therefore, it must be realized that a well defined development methodology is vital for the successful construction of a knowledge-based system as it is the case for any commercial software system. It can also be gathered from the IPS experience that there are substantial benefits to be derived from embodying a development methodology into a computer-based tool or Computer Aided Knowledge Engineering (CAKE) tool.

Based on our experience building numerous commercial KBSs we propose a development methodology specially suited for the construction of this type of software systems. This methodology has been embodied in a computer-based tool which supports knowledge engineers from the knowledge acquisition phase down to the design of the system. The tool allows for the generation of operational code as soon as any subset of the KBS's design has been completed. The architecture of

the KBSs produced by the tool reflects the principles of the underlying development methodology. These principles emphasize highly parallel and modular architectures.

The software engineering community has taught us through the Object-Oriented Approaches the importance of reusability; that is, the possibility of using and adapting off the shelves subsystems. Researchers in the AI community have tackled the problem of providing predefined problem-solving modules: for instance, Chandrasekaran's group work on generic tasks, and the KADS project. We have tried to reconcile the views of generic tasks and the KADS project together with an object oriented approach. The main objective is to provide designers of expert systems with predefined problem solving models. These models are made available in our CAKE tool to the knowledge engineer through out the design process. Our efforts have been concentrated in providing hierarchies of "canned" generic problem solving models, which we call G-TECS for generic techniques.

2. METHODOLOGICAL APPROACHES TO THE DEVELOPMENT OF KNOWLEDGE-BASED SYSTEMS

There are a number of lessons that the developers of KBSs have learned from the construction of successful and unsuccessful systems. The use of ad-hoc techniques, trial-and-error procedures, and rapid prototyping not based on sound models resulted in some disasters. These experiences have demonstrated that the construction of knowledge-based systems must be guided by a strict development methodology specially tailored for this kind of software systems. Methodologies for the construction of conventional IPS appear to be lacking some essential features required to support the development of KBSs. We believe that this is mainly due to the fact that these methodologies are mostly concerned with data modeling and functional decomposition. Moreover, these methodologies do not support activities such as knowledge modeling and knowledge processing. Therefore, no assistance is provided for tasks such as knowledge elicitation and knowledge structuring, which are essential in the construction of KBSs. This problem has been recognized by researchers in the area and some development methodologies have been proposed (14), (3), (4), (5), (6), and (1), (2), (8) A comparison between the main features of some of these approaches and ours will be presented in section 6.

The methodology we have developed is called the **Agent/Concept methodology (AC)** (9). This methodology has been embedded in a computer based tool called **EMA for Executable Methodology** for the development of knowledge-based Applications (12) and (13) due to the fact that the AC methodology is fully contained in EMA, we use the name EMA to refer to both, the methodology and the tool. EMA supports the acquisition of knowledge, the design and construction of knowledge-based applications as well as the automatic generation of executable code. It is important to notice that the applications generated by EMA reflect the underlying principles and paradigms upon which the methodology is based.

Another important aspect of EMA is the use of a predefined library of problem solving models. These problem solving models are available to the designer of KBSs through out the developing process. This will be described in detail in section 5. EMA has so far been used to develop several banking KBSs and the results obtained have been very encouraging.

3. THE KNOWLEDGE MODEL

The knowledge model is a representation of a set of *actions* and *concepts* necessary to perform some task in a certain domain. The knowledge model organizes these activities and concepts by identifying the dependencies between activities and concepts, and by recognizing the relationships among concepts. The building blocks of the knowledge model are **concepts, agents, and events**. Concepts and their relationships are organized in **structures**, which form the Information Structure. Agents are organized in **global views** and their internal functioning is described in **local views**.

3.1 CONCEPTS / INFORMATION STRUCTURES.

Concepts are used to model the **static knowledge** needed to accomplish a specific task. Concepts can represent classes of abstract objects such as *a loan, a client, or a car*. These are referred to as object-types. Concepts can also be used to refer to specific information regarding the attributes of an object-type such as *the amount of a loan, the name of a client, or the colour of a car*.

Concepts are related by relationships, the methodology distinguishes between two main classes of relationships :

a) Universal Relationships : They express relations between concepts which are permanently valid. More specifically, all the instances of the concept appearing in such a relationship must satisfy the relationship. Generalisation and Aggregation relationships belong to this class. For example, imagine a system distinguishes between two types of loans, namely commercial and personal. In this system, every instance of a loan **must** belong to either type of loan.

Generalisation (is_a / a_kind_of) : This relationship is used to express the idea that one concept could be used to describe a set of concepts with similar characteristics (subclass/class relationships). For example "a car is_a vehicle" or "A truck is_a vehicle".

Aggregation (is_part_of): Used to describe the fact that a concept can be made up of other concepts. For example, "A wheel is_part_of a car" and "an engine is_part_of a car".

b) Existential Relationships : They express relationships between instances of concepts which may (temporarily) apply. The Association-Relation belongs to this class.

Association-Relations : This is the same kind of relationship used in the Entity Relationship data models. It is used to express a special kind of relation between concepts. For instance, "a client *owns* a vehicle" this statement expresses the relationship owns between the instance of the concept client and the instance of the concept vehicle. Notice that this relationship is of the class of existential relationships since a client **might or might not** own a car.

The collection of all the modelled concepts and their relationships form the **Information Structure**. The information structure represents a map of all the static knowledge that a system contains.

As it was mentioned above, concepts represent object-types, the instances of these object-types are called **objects**. For example, in an application dealing with the scheduling of inter-banking money transfer we would find a concept like "Message 001 from Banque Cantonale Vaudoise to

Swiss Bank Corp.". This is an object which is an instance of the object-type "message". The simplest form of information in the methodology is called **information-unit** (info-unit). Info-units are made up of an object, one of its attributes and the value of this attribute. For example, "The amount of message 001 from Banque Cantonale Vaudoise to Swiss Bank Corp. is of Sfr. 15,000,000" which implies that the *attribute amount* of the *object message 001* has the *value* Sfr. 15,000,000. A generalisation of an information-unit is called an **information-type** (info-type). For example, every message has an amount. A set of info-units of the same info-type is called an **information-set** (info-set).

3.2 AGENTS

Info-sets are related among themselves by dependency relationships. This type of relationship indicates the logical dependency between one dependent info-set and one or more causal info-sets. This dependency makes explicit the info-sets necessary to produce a specific info-set and also describes the specific conditions that have to be satisfied for this to take place. This type of dependency relationship determines an agent. Agents are the active elements in the methodology. They describe the processing required to produce a dependent info-set. This processing requires different types of knowledge, namely : 1) Activation Knowledge, 2) Input Knowledge, 3) Constraint Knowledge, 4) Processing Context, 5) Functional Knowledge.

It is important to notice that agents are self-contained, they possess all the knowledge to control themselves and to process the info-sets that are sent to them. Agents do not "know" about the existence of other agents. The only communication among agents takes place through info-sets. Figure 1.0 shows how communication among agents takes via a blackboard (15).

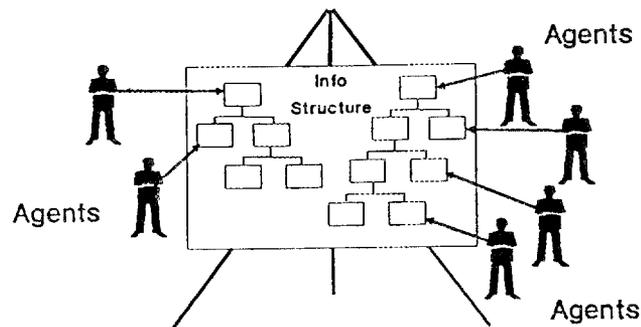


Figure 1.0 Implementation of the blackboard paradigm in the methodology.

This principle permits the systems developed with the methodology to be extremely flexible. Agents can be implemented using very distinct technologies (rule-programming, object-oriented programming, procedural programming, etc). Agents can be modified or replaced without affecting the activities of other agents.

3.3 EVENTS

An event is information or stimulus coming from the environment that surrounds an application. They represent external request for processing or external submission of information to the system. Events can be information coming from the user (**user-interface events**), or information coming from other application, or information coming directly from the domain (such as input coming from real-time sensors).

As it can be seen events are used to model the interaction of a system with the external world. The interaction includes dialogue with the user, query requests to DBs or messages coming to the system from external instrumentation.

3.4 VIEWS OF THE KNOWLEDGE MODEL

The role of a knowledge model's Agent and Information Views is to represent the system's behaviour and the knowledge structures used to produce this conduct. The views also serve as an intellectual map of an application. They put in evidence the relationships among concepts, as well as the behaviour of the application at the macro and micro level.

3.4.1 Agent Views

Agent views are divided into global and local view. The first presents a view of a system at a macro-level and the second presents a view of an application at a micro-level.

3.4.2 Information View

The information view is a representation of the information structure i.e describes all the system's concepts and all the relationships that link them. This view provides a map of all the static knowledge contained in the system.

4. A CAKE TOOL : EMA

The main goal of EMA is to support the design and development of knowledge-based systems through the use of a computer based tool. EMA encompasses the knowledge about the methodology, and as it was stated before, its final objective is to automatically generate knowledge-based applications with architectures that are akin to the principles and paradigms of the methodology. This implies that users of EMA are guided towards the construction of a systems consisting of autonomous but cooperating agents that react to stimuli.

EMA is made up of several components such as interactive graphical editors a repository, a set of guide-lines that facilitate consistency checking and an automatic generator of executable code. A knowledge engineer can use EMA to build the three different aspects of an application's knowledge model (information view and agents global and local views) by using the different specialized interactive graphical editors. The knowledge engineer is also supported by having at his disposal a library of predefined problem solving models. The user can select a specific model and tailor it to the application that he is currently implementing.

During the construction of the knowledge model some verifications are performed and warnings regarding possible inconsistencies are generated. Currently the code generator of EMA produces a skeleton of the final code, therefore this has to be manually completed through the use of a conventional editor in order to obtain a complete executable KBS.

EMA manages all the information about the application under development with the help of a repository containing all the concepts, information structures and agents views (global and local). At any given time a graphical navigator system allows the user to browse and inspect all the available relationships and a hypertext like feature allows the user to focus on a specific aspect of the application under development.

The applications generated by EMA are based on a multiagent model. This model emphasizes a highly parallel and modular organization of the components of the computer application, and it also distributes the interaction mechanisms among a set of cooperating units. Beside the modularity aspect, parallelism and distribution are interesting features for supporting interactive design without losing the objective of global systems thinking- and for implementing physically distributed applications (workstation/host cooperation is the first step towards distributed processing). This is one of our current areas of research.

EMA has been successfully used to design and develop several user-centered banking applications. Some of the systems already deployed include: Fundamental Corporate Analysis (evaluation of corporations), Credit Assessment Support System for Small and Medium Size Commercial Customers, and a system for the Assessment of Personal Loan Applications.

5. G-TECs (GENERIC TECHNIQUE).

5.1 MOTIVATIONS

The main motivation behind the idea of providing a generic description of problem solving at a high level of abstraction is to furnish knowledge engineers with predefined problem solving models that can be reused in the implementation of different systems. Therefore, the knowledge engineer can be supported from knowledge acquisition all through the actual construction of a KBS. The intent of this being that one should avoid reinventing the wheel every time one develops a new knowledge-based system. This sounds like a moral that object-oriented practitioners have been preaching to the software engineering community for a long time. We believe that the KBS field has matured enough such that we can provide libraries of problem solving models that knowledge engineers have compiled (implicitly and even explicitly sometimes) through the experienced acquired in the development of KBS.

There are multiple problems associated with the creation of any sort of library. First, the problem of what to classify must be solved. And second, a classification system must be devised. The classification of a set of problem solving models evidently involves these two problems. First, we must determine the granularity of the problem solving models. That is, with what level of detail we want to express a generic problem solving model. Second, once the granularity of the problem solving models has been determined, we must figure a way to classify these models. We must remember that for knowledge engineer to be able to navigate through a library of problem solving

models, we must devise an organization scheme that is as natural as possible to the user of the library.

5.2 DESCRIPTION

The G-TECs approach consists in providing a library of problem solving models. This has some similarities to what is referred to as **interpretation models** in the KADS project, and what is called **Generic Tasks** by Chandrasekaran's group. The differences between our approach and the two mentioned above will be presented in section 5.

In order to be able to define the G-TEC's approach we must present our conception of what problem solving entails.

The input of the problem solving process is the structure of the information that describes a problem. The processing is done by a set (one or many) of problem solving techniques. The output of the process is the information structure of a solution to a problem. For instance, we can imagine the problem of assessing the financial position of a company. The input to the problem solving process would be the structure of the financial information of the company (a hierarchical description of all the assets and liabilities of the company). The output of the process would be the judgment of the financial position of each of the areas of the company structured in some hierarchical fashion. The techniques to achieve the financial assessment of the company can vary, but they must include : a way to describe what to evaluate; and a way to describe how to evaluate it. Notice that these requirements are derived from the structure of the problem and the structure of the solution.

This very simple description of the problem solving process allows us to define and classify a set of problem solving frameworks that we have named G-TECs for Generic Techniques. It is intuitive that one can achieve a solution to a problem using many different techniques. And that a solution to a problem is linked to at least one specific technique. Based on these ideas we define two concepts for studying problem solving. The first concept is **Technical G-TEC**. For instance, optimization techniques such as simulated annealing, or search methods such as depth-first. Technical G-TECs are independent of the domain of application. The type of solution that Technical G-TECs provide and the way in which this solution is achieved is very well defined (algorithmic form). The second concept that we need to define is **Problem Specific G-TEC**. A Problem Specific G-TEC is a combination of the information structure of the solution to a problem and the specific technique used to produce the solution. The following are examples of Problem Specific G-TECs : Predicted Behaviour, Judgment, Diagnose, Plan, Design, Configuration. In EMA, G-TECs are seen as any other KBS, therefore, they are described via agents and information structures. That is, the information described corresponds to the value of attributes, the kind of general information structure it operates on and the type of actions that are allowed to be executed by the corresponding agents.

The example that it was presented above regarding the evaluation of the financial position of a company can be described with the G-TEC "Judgement". This type of G-TEC is characterised by the fact that the information described can be decomposed in hierarchies, and that partial evaluations are performed on the substructures of the hierarchies. Results are then combined to arrive to a final **judgement**. It is important to notice that the combination of the partial results is

done in an "intelligent manner" derived from the knowledge of experts in the domain. This specific knowledge would obviously vary for different domains.

6. COMPARISON BETWEEN G-TECS AND THE APPROACHES OF CHANDRASEKARAN AND THE KADS PROJECT.

We have selected to compare our approach and the one taken by the KADS project and Chandrasekaran's group (Generic Tasks or GT) because we are fairly familiar with them and also because they represent two main streams in the modeling of knowledge-based systems. There are some other approaches that are also very interesting, but for the sake of limiting the size of the paper, we have not included them

The comparison will present the differences between our approach, and GTs and KADS. In order to make the comparison as neutral and as comprehensive as possible we have selected (10) as a basis. Karbach's article presents the comparison of four approaches for the modeling of KBS. This comparison is based on three main hypothesis: 1) It is useful to describe problem solving at an abstract level; 2) Models of problem solving can be specified in a problem specific, but application independent manner; 3) Models should guide the knowledge acquisition process and aid in the structuring of the knowledge base.

6.1 HYPOTHESIS 1 : It is useful to describe problem solving on a more adequate, abstract level than that offered by general purpose knowledge representation languages.

We believe that GT and KADS provide descriptions of problem solving methods at a level of abstraction that satisfies the requirements expressed by hypothesis number one. The difficulty that we encounter is that once the abstraction of a real world problem has been done and once that this abstraction has been modelled, eventually an operational system must be produced, we have called this the abstraction-modeling-working problem. The approach taken by GT allows a knowledge engineer to achieve this goal by means of specialised environments that produce working systems. KADS, on the other hand, provides a set of interpretation models and also a language for the definition of new model. Nevertheless, KADS does not provide concrete problem solving techniques for the models that makes available to the KE.

GT solves the abstraction-modeling-working system problem in a way that we believe is not very efficient. GT's solution is not very efficient since a knowledge engineer is forced to learn each of the different environments that have been defined for each specific problem domain. In the assumption that a KE can become proficient in all these environments, he is then confronted with the problem of making all these environments communicate with each other which could be a minor problem compared with the challenge of having to integrate them into a large scale information processing system. From the software engineering point of view we believe that this process is cumbersome. KADS provides a very elaborated solution for the problem of abstracting-modeling, but it does not provide much support for the actual generation of working systems.

The approach that we have taken aims to solve the abstraction-modeling-working system by providing hierarchies of predefined G-TECS. G-TECS can be used or described at any level of abstraction along the hierarchy. The higher one moves on the hierarchy the more abstract the G-TEC becomes, and the lower one goes the more detail the G-TEC becomes (down to the

implementation level). The hierarchies of G-TECS are contained in EMA such that the automatic generation of working knowledge-based system can be supported. Figure 2.0 show a schematic representation of the abstraction and implementation mechanisms of G-TECs.

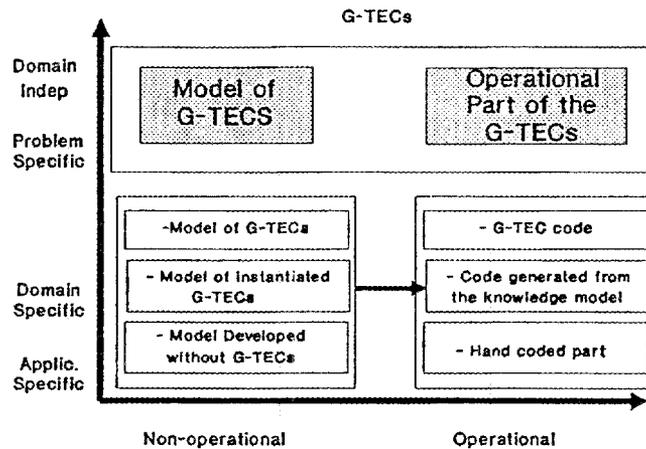


Figure 2.0 Abstraction and Implementation of G-TECS

6.2 HYPOTHESIS 2 : Models of problem solving can be specified in a problem specific, but application independent manner.

It is clear that problem solving models should be described for specific problems, but this should be independent of the instance of the problem being solved. There are substantial rewards on providing a knowledge engineer with a library of predefined problem solving models. On the other hand, it is extremely difficult if not impossible to provide a library of these methods that can satisfy the needs of every single knowledge engineer. This lead us to the requirement of **providing a formal way of describing new models**. The need for formality comes from the requirement of : combining the newly defined model with the old ones; and from the implicit requirement that the eventual automatic generation of systems must be accomplished. Both of the approaches that we have compared G-TECS with satisfy the requirements stated above in a partial manner and using different approaches.

The GT approach, provides a set of predefined problem solving models. This models have to be filled in with the actual data corresponding to the problem being modelled. Once the knowledge engineer has done this, these problem solving models are able to generate operational systems. As it can be seen this approach partially satisfy the requirements expressed above. Nevertheless, it does not satisfy the very important requirement of providing a formal manner of defining **new** problem solving models. This condition is essential since as it can be imagine knowledge engineer are very likely to be confronted with problems that require a model that has not being pre-defined.

KADS provides a set of predefined problem solving models (interpretation models). It also furnishes a formal language for defining new problem solving models. This language is very general and operates at a high-level of abstraction. This is a desirable quality since this provides high flexibility one can define new problem solving methods with a considerable ease. On the other hand, the high abstraction of the language implies in this case, a great difficulty in expressing the actual functioning of a newly defined model. This in turn, implies a tremendous difficulty for automatically generating an operational system.

The situation that we have described above it must sound extremely paradoxical. On the one hand, we want models to be specific enough such that systems can be easily implemented from this description. On the other hand, we want to be able to define new models by using an abstract and very flexible description languages. The approach that we have taken to compromise these two points of view. We believe that we should provide building block that are specific enough that can be used to precisely define a problem solving method, but that are small enough that can be combined in order to satisfy the requirement of high flexibility. The building blocks that we use for defining new G-TECs in EMA are the same that are used for defining any new knowledge-based system, namely agents and information structures. Once the new G-TEC has been defined using these building blocks, it is classified by the knowledge engineer in the corresponding level of a new or an existing hierarchy of G-TECs. At this point the G-TEC can be described in more detail by expanding the hierarchy downwards. The knowledge described at higher level of the hierarchy is available through the use of inheritance. The lowest level of the hierarchy should contain a description of a G-TEC specific enough that code can be generated from it. Notice that the KE is not obliged to provide a detail description of a G-TEC, but if he does not furnish this description, then EMA cannot automatically generate executable code for the specific G-TEC.

6.3 HYPOTHESIS 3 : Models should guide the knowledge acquisition process and aid in the structuring of the knowledge base.

The implication of this hypothesis is that the generation of operational systems based on models of problems solving should be as straight forward as possible. We believe that the most direct and precise way to perform this mapping between a model and an operational system is by, as much as possible, supporting the automatic generation of these systems. The structuring of the knowledge base is implicitly assured automatically generating the knowledge-based.

The approach taken by GT supports to some extent the requirements implied in hypotheses number three. If a knowledge engineer selects a predefined problem solving method in the GT approach, it is possible to automatically support knowledge acquisition, since the knowledge engineer is guided to filling the instances of the problem being modelled. Nevertheless, the lack of a unique procedure for defining all problem solving models implies that each GT can be structured according to a defined model, but the overall model that consist of different GTs is not necessarily structured according to any model.

KADS fulfilment of the requirements brought along by hypothesis three is amply less satisfactory than the one put forward by GT. The model for a given problem can be predefined or

defined by a knowledge engineer. Once this model has been completed and the knowledge engineer feels that the modeling process is finished, then the implementation starts. At this point KADS provides virtually no more support than advising that the final architecture of the system should reflect the model of the problem. This reflexion must be automated as it is the case in the tool that we have developed (EMA). We believe that the automatic generation of final operational systems is crucial since a great deal of the effort that was invested in correctly modeling a system could be lost if the implementation is not well controlled. Moreover, in phases such as validation, maintenance, and enhancement the needed to trace instruction in the system to parts of the model is crucial.

The approach that we take with G-TECS is to start from the requirement that the automatic generation of code is primordial to achieve any stable systems. Therefore, we believe that the knowledge engineering process must be supported by a computer based tool from knowledge acquisition to implementation. EMA furnishes this support by providing some automated aid at the knowledge acquisition phase. The modeling is fully automated and the automatic production of executable code is partially supported.

7. CONCLUDING COMMENTS

EMA is a self contained cooperative and comprehensive methodology for the development of expert systems. It is self contained because the methodology is embedded in the tool in the form of a knowledge-based system. It is cooperative because the user is seen as a partner in the development of new KBS. EMA is comprehensive because it encompasses all the information required to support the entire development cycle of an expert system. The applications generated by EMA are based on a multi-agent model. This model emphasizes a highly parallel and modular organization of the components of the computer application, and it also distributes the interaction mechanisms among a set of cooperating units. Beside the modularity aspect, parallelism and distribution are crucial features for supporting interactive design without losing the objective of global systems thinking- and for implementing physically distributed applications (workstation/host cooperation is a big step towards distributed processing).

EMA is not yet fully functional but it has gone through several versions of improvements. Several user centered banking applications have been already successfully implemented with EMA. The results obtained in the deployment of these systems have been very encouraging. Some of these systems include: Fundamental Corporate Analysis (evaluation of corporations), credit assessment support system for small and medium size commercial customers, and a system for the assessment of request of personal loans. The latest application developed with EMA is in the domain of configuration. Our current efforts are directed towards enlarging the library of G-TEChs as well as improving the automation of the support for knowledge acquisition. Regarding improvements to the methodology, we are working on providing modelling support for the integration of KBS and conventional information processing systems.

REFERENCES

1. [Breuker et al 1987] Breuker, J., Wielinga, B., VanSomeren, M., De Hoog, R., Schreiber, G., De Greef, P., Bredeweg, B., Wielemaker, J., Billaut, J. P., Davoodi, M. & Hayward, S. (1987). Model-Driven Knowledge Acquisition: Interpretation Models. Technical Report Esprit Project 1098, Deliverable Task A1. Amsterdam: University of Amsterdam, Social Sciences Informatics.
2. [Breuker & Wielinga 1989] Breuker, J., Wielinga, B., (1989). Models of Expertise in Knowledge Acquisition. In G. Guida & C. Tasso, Eds. Topics in Expert Systems Design, Methodologies and Tools. Amsterdam: North-Holland.
3. [Brown & Chandrasekaran 1989] Brown, T. & Chandrasekaran, B. (1989). Design Problem Solving: Knowledge Structures and Control Strategy, Research Notes in Artificial Intelligence. London: Pitman.
4. [Bylander & Chandrasekaran 1987] Bylander, T. & Chandrasekaran, B. (1987). Generic Tasks for Knowledge-based Reasoning: The 'right' Level of ABSTRACTION for Knowledge Acquisition. Technical Report 87-TB-KNOWAC, Technical Research Report, Ohio State University, Laboratory for Artificial Intelligence Research, Columbus, Ohio.
5. [Chandrasekaran 86] Chandrasekaran B., "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design" IEEE Expert, Vol 1, No. 3, pp 23-30, Fall 1986.
6. [Chandrasekaran 1988] Chandrasekaran, B. (1988). Generic Tasks as Building Blocks for Knowledge-based Systems. The Knowledge Engineering Review, October, 1988.
7. [Hendler 88] Hendler J.A., (ed) "Expert Systems: the User Interface", Ablex Publishing Corp., 1988.
8. [Hichman et al 89] Hichman R., Killin J. L., Land L., Mulhall T., Porter D., Taylor R. M. " Analysis for Knowledge-based Systems. A Practical Guide to the KADS Methodology" Ellis Horwood, 1989.
9. [Horvitz Probst Wenger 90] Horvitz A.I, Probst A.R, Wenger D., " An Information Engineering Methodology for the Development of Knowledge-Based Systems" Proceedings of Expert Systems 90, London, Cambridge University Press, Sept. 1990.
10. [Karbach et al 90] Karbach W., Linster M., Voss A., 1990 "Models, Methods, Roles and Tasks: Many Labels - one idea?", Knowledge Acquisition 2, Academic Press Limited, pp 279-299.
11. [Norman & Draper 86] Norman D.A., Draper S.W. "User Centered Systems Design", Lawrence Erlbaum Assoc. Publ. 1986.
12. [Probst & Wenger 90] Probst A.R., Wenger D., "Knowledge-Based Systems: Towards a design methodology which smooths away interface problems" 3rd International Symposium in Commercial Expert Systems in Banking and Insurance, Lugano Switzerland, May 15 -16, 1990.
13. [Spirgi, Probst & Wenger 90] Spirgi S., Probst A., Wenger D., 1990 "Knowledge Acquisition in a Methodology fro Knowledge-based Applications", Proceedings of IJKAW 1990, pp 382-397.
14. [Swaffield & Knight 90] Swaffield G., Knight B., "Applying systems analysis techniques to knowledge engineering" Expert Systems, Vol 7, No 2, pp 82-93, May 1990.
15. [Winston 1984] Winston, P. H., Artificial Intelligence. Second Edition. Addison-Wesley 1984.

GENERATION OF PRODUCTION RULES BY BACKWARD
SEARCH FROM NEURAL NETWORKS

Da Qun Qian, Piero Scaruffi and Dario Russi

Olivetti Artificial Intelligence Center
Olivetti Systems and Networks
Via Jervis 77, 10015 Ivrea(TO), Italy

ABSTRACT

In this paper, heuristic backward search strategies for generating production rules from neural networks proposed. By these heuristic search strategies, various types of production rules and explanations of behavior of neural networks to users can be generated.

1. INTRODUCTION

The rapid development of neural networks has attracted much attention of AI researchers. However, compared with the advantages that symbolic representations of knowledge in AI systems can be stated in a clear and relatively simple way, and can be easily documented, explained, taught and learned, neural networks have some disadvantages, i.e., the neural networks fail to offer an explanation function to the users. The users nearly always want to know why a neural network comes up with a particular answer. Sometimes they try to learn how the neural network makes decisions so that they can improve their own understanding of the problems; sometimes they want to verify that the neural network is working correctly. Therefore, by integrating neural networks with AI systems, the realization of some connectionist systems in different application areas have proven the viewpoint, i.e., the connectionist systems are more powerful than either of neural networks and AI systems. In order to overcome the above disadvantages of neural networks, this paper studies the generation of explicit representation of knowledge, namely, knowledge acquisition, from neural networks and explanation of the conclusions inferred by neural networks. In this paper, some heuristic backward search strategies for generating production rules and presenting some explanations to users are proposed.

The generation of production rules is based on the following principle idea. A neural network is viewed as a knowledge base where the knowledge is not encoded in the form in which people usually express their knowledge and which people can easily understand, such as a production rule. However, people may acquire the knowledge from the currently known behavior of the neural network. Therefore, it is possible that the production rules may be generated from this behavior. For example, assume each node in Fig. 1(a) denotes a proposition, then it may be thought that there exist some logic relationships between nodes in layer $i-1$ and nodes in layer i since the

states of the former influence the states of the latter, so that the states of the former may be regarded as premises and the states of the latter may be regarded as conclusions inferred from the logic relationships and the states of the former ($i=2,3$).

The neural network discussed in this paper has the following features:

(a). Every node denotes a proposition or a variable, i.e., a symbolic description of the proposition or the variable is attached to every node.

(b). Every node has a discrete (discontinuous) activation function, i.e., the transfer function has the following form:

$$s(k,l)=f(s) \quad (1)$$

$$s=\sum_{i,j} s(i,j)*w(i,j,k,l) \quad (2)$$

where $s(i,j)$ denotes the state of node $x(i,j)$, and $w(i,j,k,l)$ denotes the weight on the arc directly from node $x(i,j)$ to node $x(k,l)$, the i and j in $x(i,j)$ respectively denote the number of the layer and the number of the node at this layer where $x(i,j)$ is located. The formula (1) is called activation function, and formula (2) is called input function.

(c). The neural network does not contain cycles.

Some terminology on the neural network of this kind is explained first. If an arc in the neural network is directed from node $x(i,j)$ to node $x(k,l)$, then node $x(k,l)$ is said to be a successor of node $x(i,j)$, and node $x(i,j)$ is said to be a parent of node $x(k,l)$. A node in the neural network having no parent is called a root node. A node in the neural network having no successors is called a tip node. If node $x(k,l)$ is accessible from node $x(i,j)$, node $x(k,l)$ is then a descendant of node $x(i,j)$, and node $x(i,j)$ is an ancestor of node $x(k,l)$. For example, in Fig. 1(a), nodes $x(3,1)$ and $x(3,2)$ are tip nodes, $x(1,1)$, $x(1,2)$, $x(1,3)$ and $x(1,4)$ are root nodes, and they are also parents of $x(2,1)$, $x(2,2)$ and $x(2,3)$, and correspondingly $x(2,1)$, $x(2,2)$ and $x(2,3)$ are successors of $x(1,1)$, $x(1,2)$, $x(1,3)$ and $x(1,4)$, further, nodes $x(3,1)$ and $x(3,2)$ are descendants or grandchildren of $x(1,1)$, $x(1,2)$, $x(1,3)$ and $x(1,4)$, and correspondingly $x(1,1)$, $x(1,2)$, $x(1,3)$ and $x(1,4)$ are ancestors of $x(3,1)$ and $x(3,2)$. $n(i)$ denotes the number of the nodes in layer i . m denotes the number of the layers in the neural network. $X(i,j)$ denotes a symbolic description of node $x(i,j)$, which describes a variable or a proposition. A domain of a state is defined as a domain of an activation function on which the state holds, as shown in Fig. 1(b).

2. SEARCH PROCEDURES

2.1. CUT-OFF SEARCH PROCEDURE(A)

It is a backward search procedure. The backward search procedure can generate a group of production rules by which why a conclusion is inferred can be explained. It is used in the case of binary activation function (i.e. the state of a node should be in one of two possible states either s_1 or $-s_2$, $s_1 > 0$, $s_2 > 0$), but it is also easily extended to the case of multi-value

activation functions as the modified forward search procedure given in [1]. The production rules derived from the currently known information should have the minimal number of premises that is sufficient for inferring the corresponding conclusions as long as the states of nodes remain unchanged, which means that a minimal number of premises must be searched for.

(a). Let set S be equal to the node $x(k, 1)$ which is selected for explanation.

k is assigned to the number of layers of the neural network.

(b). DO WHILE $k > 1$

{

(b1) Denote all the parent nodes $x(i, j)$ of $x(k, 1)$ as set $S(1)$ which have each direct contribution for each node $x(k, 1)$ in S , i.e. $S(1) = \{x(i, j) | w(i, j, k, 1) \neq 0\}$, $l = 1, 2, \dots, n(k)$.

(b2). Classify $S(1)$ into two subsets: $S(1, 1)$ and $S(1, 2)$. $S(1, 1)$ contains all the nodes which disconfirm the state of $x(k, 1)$, and $S(1, 2)$ contains all the nodes which confirm the state of node $x(k, 1)$. $S(1, 1)$ can be deleted without affecting the state of node $x(k, 1)$. Arrange the nodes in set $S(1, 2)$ in ascending order of their contribution for the current state of successor node $x(k, 1)$, which is calculated by multiplying the weight on the arc starting from each node in $S(1, 2)$ to $x(k, 1)$ with the corresponding state of this node in $S(1, 2)$.

(b3). Delete the contributions of these nodes in $S(1, 2)$ in order arranged above until the state of node $x(k, 1)$ has been changed since $x(k, 1)$ lacks the contribution of the node being deleted so that fails to maintain the state of nodes in $S(j, 1)$.

(b4). Generate production rules from these remaining nodes in $S(1, 2)$ for node $x(k, 1)$. The rules have the following form:

$$X(i, 1)(s(i, 1))(w(i, 1, k, 1)) \text{ and } X(i, 2)(s(i, 2))(w(i, 2, k, 1)) \text{ and } \dots \text{ and } \\ X(i, t)(s(i, t))(w(i, t, k, 1)) \text{ ---> } X(k, 1)(s(k, 1))$$

where $X(i, j)$ is a symbolic description of the corresponding node $x(i, j)$; if $x(i, j)$ denotes a proposition, then $s(i, j)$ may be used as a certainty factor of node $x(i, j)$; if $x(i, j)$ denotes a variable, then $s(i, j)$ is the value of the variable; $w(i, j, k, 1)$ may be used to represent the certainty factor of $x(i, j)$ if $x(i, j)$ is a variable or the importance of $x(i, j)$ in the production rule, or the weighted logic operator AND. $(s(i, j))(w(i, j, k, 1))$ can also be replaced by $s(i, j) * w(i, j, k, 1)$ according to input function (2). $k \leq k-1$, denote these unmarked nodes in $S(1, 2)$ which are hidden nodes as set S , and mark these nodes.

}

Since there may exist several paths connecting $x(k, 1)$ and some ancestor of its, all the nodes having been expanded in this backward search should be marked in step (b4) in order to avoid to be selected for expansion again. This search procedure may not guarantee that these generated production rules have the minimal number of premises.

Example 1: The following steps show the use of the cut-off search procedure for node $x(3, 1)$ in the neural network in Fig. 2.

In step (a), $k=3$, $S=\{x(3, 1)\}$.

In step (b1), $S(1)=\{x(2, 1), x(2, 2)\}$.

In step (b2), $S(1, 1)$ is an empty set, and $S(1, 2)$ is arranged as

$\{x(2, 2), x(2, 1)\}$.

In step (b3), $x(2, 2)$ is deleted.

In step (b4), generate a production rule as follows:

$$X(2, 1)(1)(0.5) \text{-----} \rightarrow X(3, 1)(1)$$

Set $S = \{x(2, 1)\}$, $x(2, 1)$ is marked, $k=2$, go to step (b0).

In step (b1), $S(1) = \{x(1, 1), x(1, 2), x(1, 3)\}$.

In step (b2), $S(1, 1) = \{x(1, 1)\}$ and $x(1, 1)$ are deleted, and $S(1, 2)$ is arranged as $\{x(1, 2), x(1, 3)\}$.

In step (b3), $x(1, 2)$ is deleted.

In step (b4), generate a production rule as follows:

$$X(1, 3)(1)(0.5) \text{-----} \rightarrow X(2, 1)(1)$$

$k=1$, stop.

2.2. SET-BASED SEARCH PROCEDURE

In this search procedure, a new restriction is added, i.e., the number of the root nodes used in generation of production rules is required to be minimized. This restriction means that the least amount of input information is used for generating all the production rules. The set-based search procedure is a backward search procedure for generating the production rules with the minimal number of root nodes.

A set $L(i, j)$ is said to contain all the root nodes which are connected with $x(i, j)$ by some paths. The sets $L(i, j)$ ($i=2, 3, \dots, k-1$, for all j : $w(i, j, k, l) \neq 0$) should be reached at first for each node $x(k, l)$ selected for explanation. Compared with the cut-off search procedure, in this set-based search procedure the nodes are selected for expansion which only need the minimal number of root nodes to maintain their states, namely, the selection of parent nodes $x(i, j)$ [(i, j) belongs to $\{(i_1, j_1), (i_2, j_2), \dots, (i_t, j_t)\}$] of $x(k, l)$ for expansion should meet both the condition that they can maintain the state of $x(k, l)$ and the following condition:

$$| \cup L(i, j) | = \min | \cup (i, j) L(i, j) | \quad (3)$$

(i, j) = (i_1, j_1), ..., (i_t, j_t) $\cup (i, j) L(i, j)$ can maintain the state of $x(k, l)$ where $|L(i, j)|$ denotes the number of nodes contained in $L(i, j)$. The condition means that the parent nodes of $x(k, l)$ are selected for expansion towards the direction where the number of the root nodes eventually used may be minimized possibly.

Example 2: In Fig. 3, $L(3, 1) = \{x(1, j) | j=1, 2, \dots, 5\}$,
 $L(3, 2) = \{x(1, j) | j=2, 3, 4, 5\}$, $L(2, 1) = \{x(1, j) | j=1, 2, 3, 4\}$,
 $L(2, 2) = \{x(1, j) | j=2, 3, 4\}$, $L(2, 3) = \{x(1, j) | j=1, 3, 4, 5\}$, $L(2, 4) = \{x(1, j) | j=3, 4, 5\}$.
 Therefore, nodes $x(2, 2)$ or $x(2, 4)$ can be selected for expansion first according to formula (3).

2.3. CUT-OFF SEARCH PROCEDURE(B)

In this search procedure, a new restriction is added, namely, the nodes can be deleted so long as the states of the nodes selected for explanation, assuming they are tip nodes, remain unchanged. Compared with cut-off search

procedure (A), in this search procedure, the following heuristics may be used:

(a). A parent of a node can be selected for expansion if the parent node satisfies the restrictions stated in 2.2 and this section better than other parents of this node.

Example 3: In Fig. 4, in order to generate production rules that is sufficient for inferring the corresponding conclusions even if the states of the deleted nodes have changed, at first the children of tip nodes $x(3,1)$ and $x(3,2)$ are selected for expansion and there are three selections as follows:

$x(3,1)$: $x(2,1), x(2,2)$ or $x(2,1), x(2,3)$ or $x(2,2), x(2,3)$

$x(3,2)$: $x(2,2), x(2,3)$ or $x(2,3), x(2,4)$ or $x(2,2), x(2,4)$

The node group containing $x(2,2)$ and $x(2,3)$ is selected for expansion because this $x(2,2)$ and $x(2,3)$ are the common parents of $x(3,1)$ and $x(3,2)$ and only need 4 root nodes at most to maintain their states. In fact, only $x(1,3)$ and $x(1,4)$ are needed. Therefore, the generated production rules are:

$X(1,3)(1)(1)$ and $X(1,4)(1)(1)$ -----> $X(2,2)(1)(1)$

$X(1,3)(1)(1)$ and $X(1,4)(1)(1)$ -----> $X(2,3)(1)(1)$

$X(2,2)(1)(1)$ and $X(2,3)(1)(1)$ -----> $X(3,1)(1)(1)$

$X(2,2)(1)(1)$ and $X(2,3)(1)(1)$ -----> $X(3,2)(1)(1)$

(b). Then a parent of a node can be selected for expansion if it has more redundancy than other parents of this node.

Example 4: In Fig. 5, production rules can be generated as follows: $x(2,2)$ and $x(2,3)$ are selected for expansion because they are the common parents of $x(3,1)$ and $x(3,2)$, they can maintain the states of $x(3,1)$ and $x(3,2)$, and they connect the minimal number of root nodes, 4 in all. Since there exists some redundancy of $x(3,1)$ and $x(3,2)$, the following inequality describing the redundancy of $x(3,1)$ and $x(3,2)$ should be satisfied:

$$x(2,2)+2*x(2,3)>2$$

Therefore, the solutions for this inequality include:

$$x(2,2)=-1, x(2,3)=2; x(2,2)=0, x(2,3)=2; x(2,2)=1, x(2,3)=1; x(2,2)=1,$$

$$x(2,3)=2; x(2,2)=2, x(2,3)=1; x(2,2)=2, x(2,3)=2$$

Considering the constraint of ($x(2,2)=0$ or $x(2,2)=1$) and ($x(2,3)=0$ or $x(2,3)=2$) by their parent nodes, the remaining solutions now are:

$$x(2,2)=0, x(2,3)=2; x(2,2)=1, x(2,3)=2$$

From the solutions, the root nodes $x(1,4)(=1)$ and $x(1,5)(=1)$ are selected to maintain $x(2,2)=0$ and $x(2,3)=2$ or $x(2,2)=1$ and $x(2,3)=2$, i. e.,

$X(1,5)(1)(0.6)$ -----> $X(2,2)(0)$

$X(1,4)(1)(1)$ and $X(1,5)(1)(2)$ -----> $X(2,3)(2)$

or

$X(1,4)(1)(1)$ and $X(1,5)(1)(0.6)$ -----> $X(2,2)(1)$

$X(1,4)(1)(1)$ and $X(1,5)(1)(2)$ -----> $X(2,3)(2)$

(c). The backward search should satisfy the condition as follows: If the state of a node can not be maintained by the states of its parent nodes in layer 1 due to the last deletion of some its descendant node, then backtrack to some point along backward search path where the node was deleted, reserve the deleted node, start again.

3. EXTENSION OF SEARCH PROCEDURES

The above search procedures can be extended in the following cases:

3.1. EXTENSION OF NEURAL NETWORKS

(a). The input function (2) can be extended to other forms, such as, in the case of applications of neural networks in approximate reasoning, the Multiply-Add operation (2) is often replaced by a Max-Min operation, and a binary activation function is used to activate the node which has a certainty factor higher than the threshold of the binary activation function and inactivate the node which has a certainty factor lower than the threshold of the binary activation function. The search procedures proposed above can be used easily in this case.

Even if the activation function is continuous, it is also possible that these search procedures are applied if the input function is discrete. An example is the fuzzy petri net implemented by using Looney's neural network for rule-based decisionmaking [2] Looney gave an group of production rules as follows:

```

C1 and C2 ----> C4
C4 ----> C6
C5 ----> C3
C5 ----> C1
C6 ----> C' (external node)

```

and known input information(certainty factors) $C2=0.8$ and $C5=0.5$, and then used the fuzzy petri net to infer conclusions of the production rules and known input information as shown in Fig. 6 which is an illustrative graph of the fuzzy petri net. Each node $C(i)$ ($i=1,2,\dots,6$) denotes C_i in production rules and has a Max input function, and each node $N(i)$ ($i=1,2,\dots,5$) denotes a relation between premises and a conclusion of a production rule and has a Min input function. The Max and Min input fuctions are used for approximate reasoning, and the activation function employed by each node is a unit function. By using above search procedures in which the delation of nodes is replaced by decreasing values of certainty factors C_i of nodes $C(i)$ ($i=1,2,\dots,6$), the inputs of Min activation function, to the maximum, it can be seen that $C2=0.8$ can be decreased to $C2=0.5$ without affecting the previously inferred conclusions, i.e., the necessary certainty factors of premises supporting their conclusion can therefore be determined.

It is also possible to apply the search procedures to the case of both continuous input function and continuous activation function. If the state of root nodes is restricted within discrete domain, then the continuous transfer function can be regarded as discrete one since only finite values of the transfer function will be taken at each layer; otherwise, there are two strategies for generating production rules. The first one is that all the nodes are used for generating production rules since each node has its own contribution for its successor nodes so that the deletion of a node will result in the change of the states of its successor nodes. The second one is that this continuous activation function should first be made discrete. Thus, a continuous activation function may be treated equally as a discrete

activation function in above search procedures except for that each generated production rule has an error equal to the error resulting from the discretion of the continuous activation function.

(b). The search procedures can also be used in neural networks with cycles by employing the following two ideas: First, a node in a cyclic path should first be selected to be deleted so that the cycle can be broken (see Fig. 7(a)). Second, if a node in a cyclic path is reserved, then the states of other nodes in the cyclic path should be maintained; otherwise, the state of the node may be changed by the changed states of other nodes in the cyclic path so that the search has to be repeated along the cyclic path (see Fig. 7(b)).

(c). In some cases, except for tip nodes and root nodes, hidden nodes in a neural network have not any meanings, such as nodes $x(4,1)$, $x(4,2)$ and $x(4,3)$ in example 19. They are used only for realization of a mapping from input information to output information. If these nodes are included in generated production rules, the production rules will become meaningless. Therefore, the nodes should and could be deleted from the production rules.

For example, the production rules containing the meaningless nodes $x(2,2)$ and $x(2,3)$ are as follows:

$$\begin{aligned} X(1,1)(1)(2) \text{ and } X(1,2)(2)(1) &\text{ ---> } X(2,1)(2) \\ X(1,1)(1)(1) \text{ and } X(1,3)(1)(1) &\text{ ---> } X(2,2)(1) \\ X(2,1)(1)(2) \text{ and } X(2,2)(1)(1) &\text{ ---> } X(3,1)(1) \\ X(1,1)(1)(3) \text{ and } X(2,1)(1)(2) &\text{ ---> } X(3,2)(1) \end{aligned}$$

Then they can be equivalently transferred into the following form:

$$(X(1,1)(1)(2) \text{ and } X(1,2)(2)(1))(2) \text{ and } (X(1,1)(1)(1) \text{ and } X(1,3)(1)(1))(1) \text{ ---> } X(3,1)(1)$$

$$X(1,1)(1)(3) \text{ and } (X(1,1)(1)(2) \text{ and } X(1,2)(2)(1))(2) \text{ ---> } X(3,2)(1)$$

(d). Nodes in neural networks may also be used to denote other terms, such as, word, object and entity, when a neural network is used as semantic network or used in sentence processing. It is possible that the generated production rules can be used to describe the necessary and major relationships between nodes.

3.2. PARALLEL SEARCH

A major advantage of neural networks is that they can run in a parallel way. However, the generation of production rules by the above search procedures is based on a serial way. In order to realize the parallel search, the neural network should be reconstructed as follows:

(a). The computational ability of every node should further include: comparison operation, ordering operation, classification operation, solution for inequality and so on, since they are needed by every node in above search procedures.

(b). Nodes can communicate to each other by adding new arcs between these nodes for transmitting messages, since the realization of some heuristic ideas in search procedures require the communication among nodes.

(c). The neural network should be reconstructed, i.e., some nodes and arcs should be added into the neural network. Fig. 8(b) gives a typical extension

of the neural network Fig. 8(a). The added nodes A and B respectively represent the relations between $x(1,1)$ and $x(2,j)$ s($j=1,2$). Since in procedures whether a node should be deleted from some or all of production rules is determined by its corresponding success nodes, the arcs, starting from $x(2,1)$ to A and from $x(2,2)$ to B, and starting from A to $x(1,1)$ and from B to $x(1,1)$, represent the control-wires of $x(2,1)$ and $x(2,2)$ by which $x(2,1)$ and $x(2,2)$ can control the states of A and B, and eventually control the states of $x(1,1)$. The activated state of A means that the relation between $x(1,1)$ and $x(2,1)$ should be reserved in the production rule for inferring $x(2,1)$. The inactivated state of A means that the relation between $x(1,1)$ and $x(2,1)$ should be deleted from the production rule for inferring $x(2,1)$. The state of $x(1,1)$ is inactivated only when both the states of A and B are inactivated. The inactivated state of $x(1,1)$ means that $x(1,1)$ should be deleted from all the production rules. Therefore, in parallel computation, every node will either be activated or be inactivated by its successor nodes through issuing corresponding commands to it along the control-wires. Finally, the neural network will converge to a stable state, in which the state of every node, such as $x(1,1)$, A and B, stand for whether it or a relation represented by it is reserved in production rules. For example, in Fig. 8(b), if A is inactivated and B is activated, then node $x(1,1)$ is reserved in the production rule for inferring the state of $x(2,2)$ and is deleted from the production rule for inferring the state of $x(2,1)$.

Generally speaking, the realization of the heuristic ideas proposed in this paper in a parallel way needs a complex structure of the neural network.

4. RELATED WORK AND CONCLUSIONS

In the field of AI heuristic algorithms, such as, algorithm A and Alpha-beta procedure, are used in tree structure for graph search. Every node in the graph denotes a state of a database, and production rules are repeatedly used to update the state of the database under guidance of a control strategy until the state of the database matches with some goal searched for [3]. The search procedures proposed in this paper are used in neural networks for generating production rules. The heuristic search is dependent on the current state of a neural network, where states of nodes influence each other.

Gallant[4] also proposed a strategy for generating production rules from neural networks. But his strategy is restricted within a specific neural network which only permit using the activation function with three state values, (-1, 0, +1), and can only be used in a neural network of small size since the heuristics used in this strategy are very limited, otherwise the number of implicitly encoded production rules will grow exponentially with the number of node inputs. Moreover, the strategy can only generate one type of production rules. Saitoh and Nakano [5] tried to derive rules out of the causal relationship of input/output layers. But the derived rules were based on binary logic, and the certainty of rule and importance of proposition were unknown. Hayashi and Nakai [6,7] proposed a method for acquiring fuzzy inference rules from the causal relationship of input/output of neural network. The linguistic truth values included in each fuzzy proposition and

certainty of each rule can be determined by this method. However, each node is also only permitted to output three values (True(1), Unknown(0) and False(-1)), i.e., the activation function employed by each node is restricted to a three-value function, and only the causal relationship between input layer and output layer can be described by fuzzy inference rules. Bochereau and Bourguine [8,9] gave a method for solving the NP-complete problem of rule extraction from a multilayer neural network by restricting the domain of input of the multilayer neural network. However, statistical information or prior knowledge is needed in the method, the state of root nodes is restricted within boolean domain, moreover, some error may be brought about due to the use of the statistical information.

REFERENCES

1. D. Q. Qian, P. Scaruffi, and D. Russi, "Generation of Production Rules from Neural Networks by Forward Search Procedures," Proceedings of International Young Computer Scientists Conference, Beijing, China, July, 1991.
2. C.G. Looney, "Fuzzy Petri Nets for Rule-based Decisionmaking," accepted for publication in IEEE Trans. SMC, 18(1)(1988).
3. N.J. Nilsson, Principles of Artificial Intelligence. Tioga Publishing Co., Palo Alto, CA, USA (1980).
4. S. I. Gallant, "Connectionist Expert Systems," accepted for publication in Comm. of ACM, 31(2)(1988).
5. K. Saitoh and R. Nakano, "Medical Diagnostic Expert System Based on PDP Model," Proceedings of IEEE Conference on Neural Networks, 1988, Vol. 1, 255.
6. Y. Hayashi and M. Nakai, "Automated Extraction of Fuzzy Production Rules Using Neural Networks. Proceedings of 5th Fuzzy Systems Symposium, 1989, 169
7. Y. Hayashi and M. Nakai, "Automated Extraction of Fuzzy IF-THEN Rules using Neural Networks," accepted for publication in Trans. IEE Japan, 110-C(3)(1990).
8. L. Bochereau and P. Bourguine, "Rule Extraction and Validity Domain on a Multilayer Neural Network," Proceedings of IEEE and INNS Joint Conference on Neural Networks, San Diego, 1990, Vol. 1, 97.
9. L. Bochereau and P. Bourguine, "Extraction of Semantic Features and Logical Rules from a Multilayer Neural Network," Proceedings of IEEE and INNS Joint Conf. Neural Networks, Washington, DC, 1990.

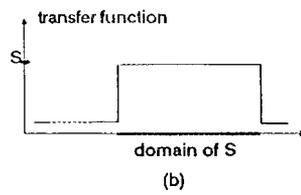
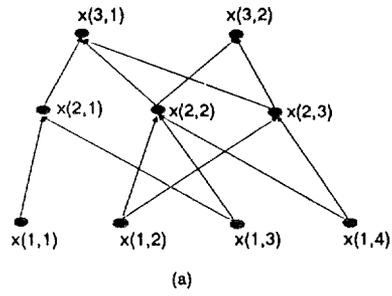


Fig. 1 Neural network and transfer function

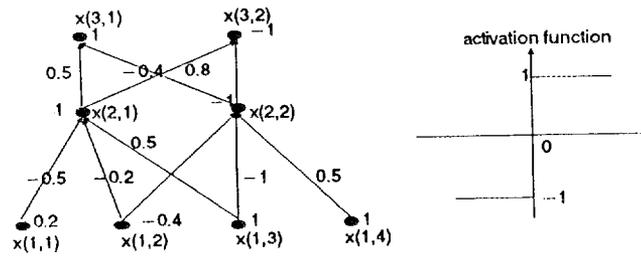


Fig. 2 Example 1

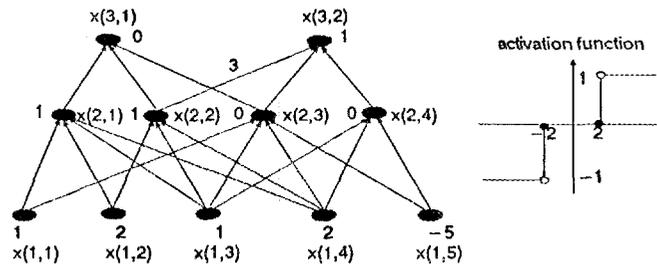


Fig. 3 Example 2

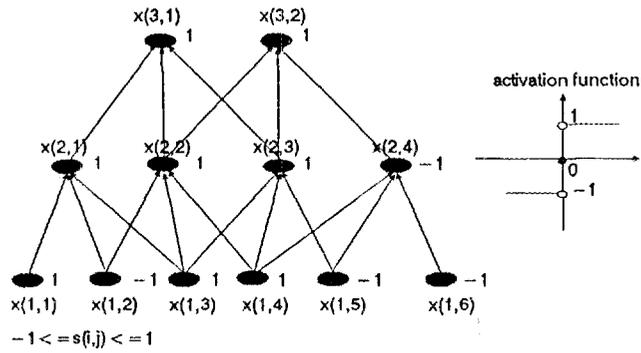


Fig. 4 Example 3

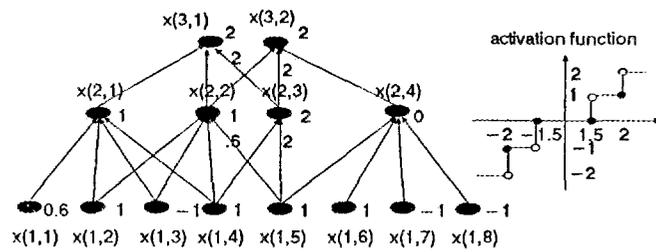


Fig. 5 Example 4

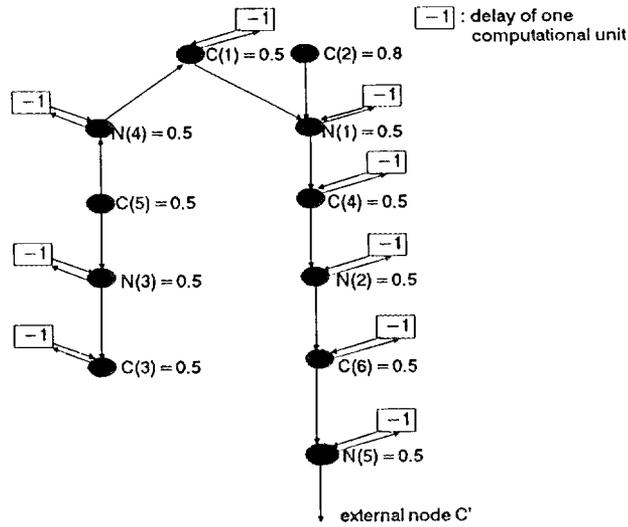
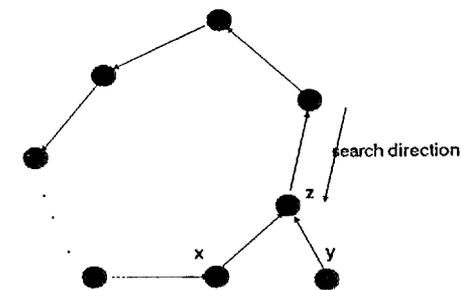
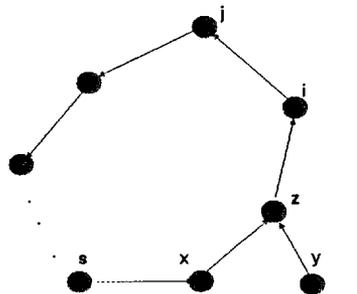


Fig. 6 A fuzzy petri net implemented by using a neural network for rule – based decisionmaking

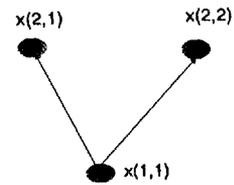


(a) The cycle can be broken if node x is deleted and node y is reserved in backward search

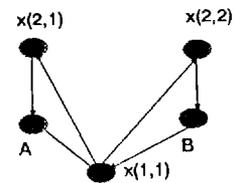


(b) The state of node z can be changed by changing the states of other nodes i, j, ..., s, x

Fig. 7 Treatment on cyclic path



(a)



(b)

Fig. 8 Parallel search

Computational Nonmonotonism and The Qualification Problem

Arcot Rajasekar

Department of Computer Science
University of Kentucky, Lexington, Kentucky 40506

Abstract

Human beings isolate qualification checking from general problem solving and do not check for all qualifications all the time. But, they are prepared to allow erroneous conclusions in favour of exhaustive checking. A strategy which has these human traits can be accepted as a possible solution for the qualification problem. In this paper we develop a strategy, called computational nonmonotonism (CN), which mimics the above paradigm. In a CN system a qualification is applied only if it is relevant to a context or if it has a priority greater than a certain level warranted by the situation or scenario. The conclusions from a scenario can be different from another because of different qualifications which become applicable depending on the scenario. We apply computational nonmonotonism to a system of rules and qualifications and provide declarative and operational semantics for inferring from the system. We also show the generality of the approach, by applying the computational nonmonotonism technique to default reasoning and circumscription. The three applications capture differing intuitions of computational nonmonotonism. We also point out that the approach can be adapted for real-time, time-constrained nonmonotonic reasoning.

1 Introduction

One of the difficult problems in common sense reasoning is the qualification problem: the problem of checking for abnormality conditions which can invalidate a conclusion. There are two aspects to the qualification problem. One aspect is that it is an important problem that needs to be solved in common-sense reasoning since it provides a basis for nonmonotonic reasoning. In another respect it is a difficult problem to solve since in real world situations there can be numerous qualifications that need to be checked before validating a conclusion. Human beings seem to tackle the two aspects very well, they are able to make nonmonotonic inference without being unduly bogged down with checking for numerous qualifications. The method that human beings seemingly employ is that *they isolate qualification checking from general problem solving and do not check for all qualifications all the time. But, they are prepared to allow erroneous conclusions in favour of exhaustive checking.* In real life, we might check for some (obvious) qualifications, but may not spend that much time for checking for obscure qualifications, which might actually annul our conclusions. When hard pressed for time, we even ignore to check obvious qualifications. That is, humans, even though they can deduce information which might

contradict their conclusions, might conclude erroneously, due to lack of time, space (fatigue) and or other reasons. A strategy which has these human traits can be accepted as a possible solution for the qualification problem. In this paper we study a strategy, called computational nonmonotonicism, which mimics, in some fashion, the above paradigm. We apply the concept of computational nonmonotonicism to three nonmonotonic reasoning systems: rules with exceptions [3], default reasoning [12] and circumscription [5]. Each of these applications capture a different aspect of computational nonmonotonicism.

The qualification problem was first identified by McCarthy [4] in the context of the missionaries and cannibals puzzle. He described circumscription [5] as a paradigm for solving the qualification problem. Since then, the qualification problem has been investigated in the framework of nonmonotonic reasoning and several paradigms have been introduced. Some of the techniques that address the qualification problem include default reasoning [12, 11, 3], modal-based logic [10], inheritance theory [2], temporal reasoning [13], reasoning about action [1], etc.,. In existing artificial intelligence systems, the qualification problem is solved by encoding qualifications as part of the theory and finding an extension which minimizes the conclusions that can be inferred from the theory. Various minimization policies are employed which lead to different reasoning paradigms and different sets of conclusions.

The problem of numerous qualifications is solved, by these systems, by ignoring several qualifications and encoding only a few relevant ones. That is, the number of qualifications that are checked are limited but the limits are applied *a priori* through encoding. The encoded qualifications are checked in all situations, every time. This might lead to unnecessary checking for inapplicable qualifications. For example, consider that we know that tweety is a tropical bird and we want to find out if it flies. Even in this case, tweety is checked to see if it is a penguin. Such checking is unnecessary since, by context, tweety cannot be a penguin. The reason for the unnecessary checking done in existing systems is due to the fact that the qualification which need to be checked are interlocked with general problem solving and hence gets checked every time.

In this paper, our approach is not to provide a different representation schema for the qualification problem from those suggested in the literature. Instead, we provide an effective computational means to address the qualification problem in a real-life situation, where the number of qualifications to be checked might overwhelm a system and where every qualification need not be checked in all situations. The distinguishing features of computational nonmonotonicism is that, the qualification problem is considered in isolation from the general causality-based problem solving, because of which the processing of qualifications proceeds independent of the general problem solving and can be controlled by considerations such as priority, context, time and space. That is, not all qualifications will be checked but only those which are of high priority or those that are relevant to the context, or those which can be checked within certain time and space limits. Because of this, our approach also shares with human reasoning, the property of being fallible: that is, it may make unsound conclusions. But we show that, within the limits of the above constraints the conclusions made by the system is sound. We make this concept of soundness clear later in the paper.

The motivation behind our approach is to control the qualification problem and to provide answers to queries within some realistic constraints. We call our approach as *computational nonmonotonicism (CN)* since changing the context or scenario in which a query is asked changes the answer to that query.

In this paper we provide a syntax and a semantics for a computational nonmonotonic system by applying it to three nonmonotonic reasoning systems. We discuss, in detail, the case of rules and exceptions [3] and develop declarative and operational semantics for a computational nonmonotonic system. We also show how the technique of computational nonmonotonicism can be applied to other nonmonotonic reasoning systems such as default reasoning [12] and circumscription [5, 6]. In this paper we consider only computational nonmonotonicism achieved through context and priority-control and do not cover time and space limited nonmonotonicity.

2 Rules and Exceptions

The semantics of a system of rules and exceptions have been developed by Poole [11] and Sadri and Kowalski [3]. We modify the semantics of [3] to provide a *computational nonmonotonic (CN) system* of rules and exceptions. The system consists of two types of sentences: rules and qualifications. The *rules* are of the form

$$(1) \quad A_0 \leftarrow A_1, \dots, A_n$$

where $n \geq 0$, and the A_i s are atoms. The *qualifications* are of the form

$$(2) \quad [N, M] \neg A_0 \leftarrow A_1, \dots, A_n$$

where $n \geq 0$, the A_i s are atoms and $N, M \geq 0$. The number N denotes a *priority number* and captures the priority of the qualification with respect to other qualifications (the smaller the priority number then higher is its priority). The number M denotes a *context number* which defines the context in which the qualification applies. Note that there are no negative literals in the antecedent of (1,2). The reason being that such negative antecedents can be encoded as qualifications with $N = M = 0$. The conclusions made from the rules and the qualifications depend upon a particular situation, called scenario.

Definition 2.1 A *scenario* S is a four-tuple $\langle D, Q, M_S, N_S \rangle$, where D is a set of rules, Q is a set of qualifications, N_S and M_S are two system-wide numbers denoting respectively the priority number and the context number of the scenario. \square

A qualification gets checked in a scenario S if its priority number $N \leq N_S$ and if its context number $M = M_S$. There are two special cases. When $M = 0$ the qualification always gets checked provided the priority is not lower than the system-wide priority. That is, a qualification with $M = 0$ can be seen as a default qualification which gets checked always independent of the context. Similarly when $M_S = 0$ every qualification in the scenario gets checked independent of their individual context number. This can be used when one wants to provide an answer which is correct in all contexts. By assigning a very large value to N_S and having $M_S = 0$ the system degrades to a traditional rules and exceptions system where every default rule and every exception becomes applicable. The use of context sensitive qualifications can be illustrated as follows.

Example 2.1 Consider a plan generation system implemented using the CN system. There can be two possible scenarios: a fair weather scenario and a snowing scenario. One can query the system to generate a plan for reaching from A to B. If the weather is fine, the fair weather scenario can be chosen and a plan is generated which complies to certain qualifications. If it is snowing, the scenario in the system can be changed to a snowing scenario and a different plan is generated, since a new set of qualifications, such as using only snow emergency routes and avoiding steep-hill climbing, become appropriate. These new qualifications are irrelevant in a fair weather scenario and their checking should be avoided. \square

The use of priorities for checking qualifications can be illustrated with a planning example.

Example 2.2 Consider a fair-weather scenario in the example above. There can be two qualifications which can constrain a plan as follows:

Q_1 : plan is void if there is construction on road from A to B

Q_2 : plan is void if there is a traffic backup on road from A to B

Let the priority-numbers of Q_1 and Q_2 be 1 and 2 respectively.

Now, consider that one is doing high-level planning and checks only the qualifications which are of high priority (say $N_S = 1$). Then, only Q_1 is checked. If there is a road from A to B which is not under construction, the plan is approved. Consider another scenario, where one is doing low-level planning and even low level qualifications need to be checked. Then, one can query the system with (say) $N_S = 5$ and might get a different answer (to that given for high level planning) depending upon whether the road is having a traffic backup or not. Hence, prioritization of qualifications allows one to choose the level of detail (or risk) one is willing to consider as appropriate to the scenario. \square

In order to find the set of applicable rules and qualifications in a particular scenario we define a belief base as below:

Definition 2.2 A *belief base* of a scenario $S = \langle D, Q, M_S, N_S \rangle$ is a set $D \cup Q'$ where $Q' = \{ \neg A_0 \leftarrow A_1, \dots, A_n \mid [N, M] \neg A_0 \leftarrow A_1, \dots, A_n \in Q, \text{ and } N \geq N_S \text{ and } (M = 0 \text{ or } M = M_S \text{ or } M_S = 0) \}$ \square

Example 2.3 Consider a planning program written in CN language.

The default rules D are:

$\{ \text{path}(A, B) \leftarrow \text{connected}(A, B); \text{path}(A, B) \leftarrow \text{connected}(A, C), \text{path}(C, B); \text{connected}(la, sf); \text{nonsnowemergncyrt}(la, sf); \text{hassteepslopes}(la, sf) \}$

The qualifications Q are:

$\{ [1, 20] \neg \text{connected}(A, B) \leftarrow \text{construction}(A, B); [2, 20] \neg \text{connected}(A, B) \leftarrow \text{trafficbackup}(A, B); [1, 30] \neg \text{connected}(A, B) \leftarrow \text{nonsnowemergncyrt}(A, B); [3, 30] \neg \text{connected}(A, B) \leftarrow \text{hassteepslopes}(A, B) \}$

20 and 30 denote fair-weather and snow-weather contexts resp.

Belief base of $S_1 = \langle D, Q, 1, 20 \rangle$ is given by

$$B_1 = D \cup \{ \neg \text{connected}(A, B) \leftarrow \text{construction}(A, B) \}$$

Belief base of $S_2 = \langle D, Q, 2, 30 \rangle$ is given by

$$B_2 = D \cup \{ \neg \text{connected}(A, B) \leftarrow \text{nonsnowemergncyrte}(A, B) \}$$

□

3 Semantics of CNR systems

We first define an extension which provides a declarative meaning of a scenario in a computational nonmonotonic reasoning system using rules and qualifications. First, we need the notion of a Herbrand base of a belief set. The Herbrand Base of a belief set B , denoted as $HB(B)$, is the set of all ground atoms that can be formed using the predicate symbols, function symbols and constant symbols that appear in B . An extension is a set of atoms which can be derived from the rules of the scenario and which is consistent with the belief set of the scenario.

Definition 3.1 Let S be a scenario and $B = D \cup Q$ be its belief set. Then an *extension* of S , $E(S)$, is the smallest subset of $HB(B)$ such that for any clause $A \leftarrow A_1, \dots, A_n$ in D , if $A_1, \dots, A_n \in E(S)$ and $A \notin Q(S)$ then $A \in E(S)$.

The qualification set of S , $Q(S)$, is the smallest subset of $HB(B)$ defined as follows:

for any clause $\neg A \leftarrow A_1, \dots, A_n$ in Q , if $A_1, \dots, A_n \in E(S)$ then $A \in Q(S)$ □

The definition of extensions can be seen as a modification of answer sets defined in [3].

Example 3.1 Considering scenarios S_1 and S_2 from Example 2.3,

$$E(S_1) = \{ \text{connected}(la, sf), \text{nonsnowemergncyrte}(la, sf), \\ \text{hassteepslopes}(la, sf), \text{path}(la, sf) \}$$

$$E(S_2) = \{ \text{nonsnowemergncyrte}(la, sf), \text{hassteepslopes}(la, sf) \}$$

□

Next, we provide a procedural method for computing from a CN system.

Procedure 3.1 CN Procedure Given a scenario: $S = \langle D, Q, N_S, M_S \rangle$, to find if R is *true* in S , where R is a set of atoms

recursive solve(R)

$L_1 = \text{true}$

while $R \neq \emptyset$ **and** $L_1 = \text{true}$

{ A is an element in R

$R = R - \{A\}$

$T =$ set of rules in D such that

$A' \leftarrow A_1, \dots, A_n \in D$ **and** $A = A'$

$L_2 = \text{false}$

while $T \neq \emptyset$ **and** $L_2 = \text{false}$

{ $A' \leftarrow A_1, \dots, A_n$ is an element in T

$T = T - \{A' \leftarrow A_1, \dots, A_n\}$

$L_2 = \text{solve}(\{A_1, \dots, A_n\})$ } **end while**

```

    if  $L_2 = true$  then  $L_1 = qualify(A)$ 
    else  $L_1 = false$  } end while
return  $L_1$ 
recursive qualify( $A$ )
   $T =$  set of qualifications in  $Q$  such that
     $[N, M] \neg A' \leftarrow A_1, \dots, A_n \in Q$  and  $A = A'$ 
    and  $N \leq N_S$  and  $(M = M_S$  or  $M = 0$  or  $M_S = 0)$ 
   $L = false$ 
  while  $T \neq \emptyset$  and  $L = false$ 
    {  $[N, M] \neg A' \leftarrow A_1, \dots, A_n$  is an element in  $T$ 
     $T = T - \{[N, M] \neg A' \leftarrow A_1, \dots, A_n\}$ 
     $L = solve(\{A_1, \dots, A_n\})$  } end while
  if  $L = true$  then return false
  else return true

```

□

Example 3.2 Consider Example 2.3. Let the query be to find if $path(la, sf)$ is *true* in scenario S_2 . A computation of the CN procedure can be:

- (1) $solve(\{path(la, sf)\})$
- (2) $solve(\{connected(la, sf)\})$
- (3) $solve(\{\})$ returns *true*
- (4) $qualify(\{connected(la, sf)\})$
- (5) $solve(\{nonsnowemergncyrtel(la, sf)\})$
- (6) $solve(\{\})$ returns *true*
- (7) $qualify(\{nonsnowemergncyrtel(la, sf)\})$ returns *true*
- (8) *true* is returned for $solve(\{nonsnowemergncyrtel(la, sf)\})$
- (9) *false* is returned for $qualify(\{connected(la, sf)\})$
- (10) *false* is returned for $solve(\{connected(la, sf)\})$
- (11) steps (2) to (10) get repeated for $solve(\{connected(la, C), path(C, sf)\})$ and returns *false*
- (12) *false* is returned for $solve(\{path(la, sf)\})$

The query fails and $path(la, sf)$ is not true in scenario S_2

□

The following theorem shows the equivalence between the extensions of a CN system of rules and qualifications and the answer set generated using the CN procedure.

Theorem 3.1 Soundness and Completeness of CN Procedure

Let S be a scenario whose belief set is propositional. Then, a ground atom A is in $E(S)$ if and only if the query $solve(A)$ to the CN procedure returns the value *true*.

Proof Sketch: From the definition of $E(S)$ it can be seen that A is *true* if and only if it matches the head of a rule in D provided (1) the body is also *true* and (2) A is not disqualified by Q . The recursive call of $solve(\{A_1, \dots, A_n\})$ in the procedure **solve** takes care of condition (1). Condition (2) is taken care of by the call $qualify(A)$ which should succeed to meet condition (2). From the definition of $Q(S)$ it can be seen that $\neg A$ is

not in $Q(S)$ if and only if all proofs of $\neg A$ from $D \cup Q$ fail to succeed. The *while* loop in the procedure **qualify** tests for all such proofs for $\neg A$ by testing if the bodies of all the matching qualification rules fail. Since, we are dealing with a propositional belief set, calls made to procedures **solve** and **qualify** are decidable. Hence, $\text{solve}(A)$ returns *true* if and only if $A \in E(S)$. \square

4 CNR and other NMR paradigms

In the preceding sections we provided a general approach to computational nonmonotonicism in the context of a system of rules and exceptions. Even though we described a particular syntax and semantics the notions embedded in computational nonmonotonicism can be easily transported to other nonmonotonic reasoning systems to provide a richer nonmonotonic capability with a facility to reason under different situations based on context and priority. First, we show how we can extend Reiter's default reasoning to provide computational nonmonotonicism.

4.1 Default Reasoning

We consider a restricted form of Reiter's [12] default reasoning. Reiter's default theory consists of a set of formulas and a set of defaults. Computational nonmonotonicism can be achieved by restricting the application of defaults depending upon the context and priority. A *CN-default theory* is a four tuple $\langle W, D, N_S, M_S \rangle$ where W is a set of well formed formulas, N_S and M_S are two system-wide parameters defining context and priority as shown in Section 2. D is a set of defaults of the form:

$$\frac{[N, M] \alpha : \mathcal{M}\beta_1, \dots, \mathcal{M}\beta_n}{\gamma}$$

The above formula means that if α is proven and each β_i is consistent then γ can be inferred provided the default rule is not restricted by system-wide parameters of context number M_S and priority number N_S . A default rule is applicable if its context number $M = M_S$ or if $M = 0$ (see Section 2 for explanation) and when $N \leq N_S$. The consistency of each β_i can be taken as that $\neg\beta_i$ is not provable from the system in a particular scenario. In this paper, we are only considering the propositional default theory.

We next define the expansion computed using a CN-default theory.

Definition 4.1 Let $S = \langle W, D, N_S, M_S \rangle$ be a CN-default theory. A *CN-default extension* of S is defined as the smallest set satisfying the following properties:

- (1) $W \vdash \gamma$ then $\gamma \in E(S)$
- (2) If there is a CN-default rule of the form given above, then
 if $\alpha \in E(S)$ and $\forall i, n \geq i \geq 1, \neg\beta_i \notin E(S)$
 and $N \leq N_S$ and $(M = M_S$ or $M = 0$ or $M_S = 0)$
 then $\gamma \in E(S)$

The following example illustrates the approach of CN-default theory.

Example 4.1 Consider the Nixon diamond problem: Most quakers are pacifists. Most republicans are nonpacifists. Nixon is a quaker and a republican. The problem does not allow any inference regarding Nixon's pacifism. But let us consider two scenarios as part of the problem. The statement that 'most quakers are pacifists' applies only when we are dealing with mild-pacifists and the statement that 'most republicans are nonpacifists' applies only when we are dealing with hawkish-republicans. The extended problem can be encoded as the following CN-default theory:

$$W = \{quaker(nixon), republican(nixon)\}$$

$$D = \left\{ \frac{[0,1] quaker(X) : \mathcal{M} pacifist(X)}{pacifist(X)}, \frac{[0,2] republican(X) : \mathcal{M} \neg pacifist(X)}{\neg pacifist(X)} \right\}$$

Consider scenario $S_1 = \langle W, D, 0, mild_quakers \rangle$. Then,

$$E(S_1) = \{pacifist(nixon), quaker(nixon), republican(nixon)\}$$

Next, consider scenario $S_2 = \langle W, D, 0, hawkish_republicans \rangle$. Then,

$$E(S_2) = \{\neg pacifist(nixon), quaker(nixon), republican(nixon)\}$$

Finally, considering a scenario, $S_3 = \langle W, D, 0, 0 \rangle$, where all defaults apply, we get either $E(S_3) = E(S_1)$ or $E(S_3) = E(S_2)$ since both the defaults become applicable in S_3 . \square

By restricting the application of particular default rules, extensions are generated which correspond to different scenarios. For example, scenario given by S_3 reduces the problem to default reasoning as defined by Reiter [12].

4.2 Circumscription

Applying the concept of computational nonmonotonicity was natural in the case of rules with qualification and in the case of default reasoning. The reason for this is that the qualifications and defaults are isolated and can be controlled through system wide context and priority relationships. In the case of circumscription, achieving computational nonmonotonicity is not so easy since the theory is a monolith and is acted upon by the circumscriptive schema as a whole. But, one method suggests itself, that of controlling the circumscriptive minimization of certain predicates depending upon the scenario. This is an extension of the concept of *protection* introduced by Minker and Perlis [8, 9, 7]. That is, we can use scenarios to inhibit the minimization of some of the predicates. The value of these predicates will be inhibited from being considered *false* due to circumscription. We can motivate computational circumscription using the following example.

Example 4.2 [9] Someone asks whether you have ever known the phone number of a movie star. You pause only split seconds before answering 'No'. Later, on being asked whether you ever known the phone number of your uncle in Chattanooga, you hesitate, frown, and end up saying that you are not sure. We apparently circumscribe on a movie star's phone number, but not on a relative's. In this case, we 'protect' the answer from taking on a value 'No'. That is, certain things are circumscribed whereas the uncertainty of certain other things are protected. \square

In the above example, we can ascribe two contexts, one, called *moviestars-context*, where we allow the circumscription on a phone number and another, called *relatives-context*, where no circumscription is allowed on a phone number. We make precise the idea of restricted circumscription by extending the definition of protected circumscription.

A *computationally protected set* is a set of elements of the form:

$[N, M] p_i$, where p_i is a predicate (possibly instantiated) and N is its priority number and M is its context number.

Definition 4.2 A *computationally protected theory* is a four tuple defined by a scenario S , $A = \langle T, R, N_S, M_S \rangle$ where T is a set of well formed formulas, R is a computationally protected set of predicates and N_S and M_S are two system-wide parameters defining context and priority of the scenario S , as in the case of the systems defined in Sections 2 and 4. \square

In the definition below EP_S stands for the set of predicates which are protected in a given scenario S defined by system-wide parameters M_S and N_S . Given a computationally protected theory $A = \langle T, R, N_S, M_S \rangle$,

$EP_S = \{p_i \mid [N, M] p_i \in R \text{ and } N \leq N_S \text{ and } (M = M_S \text{ or } M = 0 \text{ or } M = 0)\}$.

That is, EP_S -things are protected in S .

Definition 4.3 (modified from [7]) Let T be a theory and EP_S be a protected set defined by a scenario S . Let P and Z be two disjoint set of predicates where the predicates in P are circumscribed and those in Z are allowed to vary. Then the *computationally protected circumscription schema* is

$$CIRC(T; P; EP_S; Z) = \\ T(P, Z) \wedge \forall P', Z' ((T(P', Z') \wedge P'/EP_S \Rightarrow P) \Rightarrow P'/EP_S \Leftrightarrow P)$$

where P' and Z' are sets of predicate symbols similar to P and Z . Notation T/U denotes $T \& \neg U$. \square

The following example shows how scenarios can be used to protect predicates from being minimized through circumscription.

Example 4.3 Consider the following encoding of Example 4.2.

$T = \{know_number(X) \leftarrow know_person(X) ; know_person(mother) ; \\ person(uncle) ; person(filmstar) \}$

Let $R = \{[0, 2] know_number(uncle)\}$. That is, predicate $know_number(uncle)$ is protected when $N_S \leq 0$ and ($M_S = 2$ or $M_S = 0$). Consider the scenario S where $A = \langle T, R, 0, 2 \rangle$. Then, $EP_S = \{know_number(uncle)\}$. Applying computationally protected circumscription for $P = \{know_number\}$, $Z = \{know_person\}$, we get

$CIRC(T; P; EP_S; Z) = T \cup \{\neg know_number(filmstar)\}$.

This allows the inference of $\neg know_number(filmstar)$, whereas the question *whether you ever knew the phone number of your uncle* does not produce a negative answer, since the predicate $know_number(uncle)$ is not minimized in scenario S . The predicate $know_number(mother)$ is *true* in all scenarios. \square

5 Discussion

Computational nonmonotonicism provides a facility to make conclusions depending upon the situation or scenario. Nonmonotonicism occurs because of the changing context and the changing priority level of the system. We have discussed computational nonmonotonicism in the light of three systems: rules with exceptions, default reasoning and protected circumscription. The CN theory of rules and qualifications extends the semantics of rules and exceptions [3] to include context and priority. Our application of CN theory to default reasoning can be seen as an extension of Poole's default reasoning system [11]. He applies constraints to defaults, but the constraints are applied all the time with no notion of context or priority. Our application of CN theory also extends the concept of protected circumscription [9] to scenarios. The three cases capture different aspects in their application of the CN paradigm. CN theory applied to rules and exceptions provides a means to trim and/or enlarge an expansion depending upon the applicable set of qualifications. CN-default theory provides a method to restrict the application of certain default rules and thereby controlling the default assumptions that can be made from the system. CN-default theory provides a method for choosing one extension among multiple extensions depending upon the situation. In the case of circumscription, the CN technique allows one to control the set of predicates which is minimized by circumscription. We have shown the utility of computational nonmonotonicism by applying it to several nonmonotonic reasoning paradigms.

The approach developed in this paper can be easily adapted for real time applications where one needs to provide results within certain time constraints. Because of the way the knowledge base is bifurcated, as rules and qualifications, the rule-processing and qualification-checking can proceed in parallel. The number of qualifications that are checked is a function of the time available. If the amount of time given is short, the number of qualifications that get checked is also small and the answer given is consistent only with the qualifications checked. When more time is given, a larger number of qualifications get checked and the answer provided is 'more correct' compared to the answer given with less qualification processing. This type of processing provides yet another type of computational nonmonotonicism. The approach can also be generalized by providing context (or situation) information for individual rules in the CN-system of rules and exceptions, and for individual axioms in W in a CN-default theory.

References

- [1] M.L. Ginsberg and D.E. Smith. Reasoning about Action II: The Qualification Problem. *Artificial Intelligence*, 35:311–342, 1988.
- [2] J. Horty, R. Thomason, and D. Touretzky. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. Technical Report CMU-CS-87-175, Computer Science Department, Carnegie Mellon University, 1987.

- [3] R.A. Kowalski and F. Sadri. Logic Programs with Exceptions. In D.H.D. Warren and P. Szerdi, editors, *Proceedings of ICLP90*, pages 598–613, 1990.
- [4] J. McCarthy. Epistemological Problems in Artificial Intelligence. In *Proc. 5th International Conference on Artificial Intelligence*, pages 1038–1044, 1977.
- [5] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1 and 2):27–39, 1980.
- [6] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Proc. AAAI Workshop on Non-Monotonic Reasoning*, pages 295–323, 1984.
- [7] J. Minker and D. Perlis. Computing protected circumscription. *Journal of Logic Programming*, 2(4):235–249, December 1985.
- [8] J. Minker and D. Perlis. Applications of protected circumscription. *Proc. Conference on Automated Deduction*, May 1984.
- [9] J. Minker and D. Perlis. Protected circumscription. *Proc. Workshop on Non-Monotonic Reasoning*, pages 337–343, October 17-19, 1984.
- [10] R. C. Moore. Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence*, 25(1):75–94, 1985.
- [11] D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence*, 36:27–47, 1988.
- [12] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1 and 2):81–132, April 1980.
- [13] Y. Shoham and D. McDermott. Problems in Formal Temporal Reasoning. *Artificial Intelligence*, 36:49–61, 1988.

TASK ALLOCATION BY A TEAM OF LEARNING AUTOMATA

Franciszek Seredynski ¹

Institute of Computer Science, Polish Academy of Sciences
00-901 Warsaw PKiN, P.O.Box 22, Poland

Abstract

Parallel algorithm of static allocation of a program into parallel computer is proposed. The algorithm is seen as a sequence of games of learning automata team migrating in a computer system graph.

1. INTRODUCTION

As computer architectures evolve towards massive parallelism, a major research question that has developed is the question of assigning modules of a program into a parallel computer for maximum performance. The problem belongs to a class of combinatorial optimization problems and is known as NP-complete. Various techniques have been treated lately for possible solutions. These solutions are based on application of mathematical programming [1], graph theory [2], branch and bound algorithms [3], or queuing theory [4].

Allocation algorithms can be generally classified into static allocation algorithms [2, 5] and dynamic load balancing algorithms [6, 7]. While load balancing algorithms are often parallel and distributed, static allocation algorithms are typically sequential and represent a bottleneck in execution on a parallel machine. In this paper we concentrate on working out parallel and distributed algorithms of static allocation of program graphs in message passing multiprocessor systems.

Stochastic search techniques such as genetic algorithms [8, 9, 10] or Boltzmann Machine [11, 12] modelling biological mechanisms existing in nature have been applied lately to difficult problems of combinatorial optimization. We propose in the paper another biologically motivated technique using a concept of a learning automaton [13, 14, 15, 16, 17] and based on self-organizing features of learning automata teams [18, 16, 17].

In the paper Section 2 contains a computational model and a structure of an allocation algorithm. Section 3 provides a theoretical background for learning automata team models. In Section 4 an algorithm of static allocation as a sequence of dynamic automata games is presented. Section 5 discusses the algorithm and presently available results. Section 6 makes some concluding remarks.

¹Currently at Laboratoire de Genie Informatique, IMAG B.P.53x-38041-Grenoble cedex, France

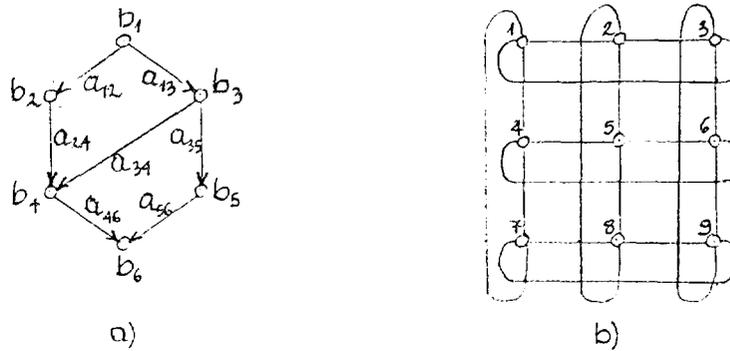


Figure 1: Examples of (a) a program graph, (b) a system graph

2. COMPUTATIONAL MODEL AND A STRUCTURE OF ALLOCATION ALGORITHM

As a general model of a distributed computer system we employ the system graph $G_s = \langle V_s, E_s \rangle$, where vertices V_s represent the nodes of multiprocessor system containing N_s identical processors and edges E_s represent the interconnection pattern of the system. As a general model of a parallel program we use a directed weighted graph $G_p = \langle V_p, E_p \rangle$ with set V_p of N_p nodes representing program modules and with collection of arcs E_p representing connections between modules. Fig.1 shows examples of a program graph and a system graph.

To develop a distributed representation of the system graph for the purpose of the allocation algorithm our structural description of v_s^i system node ($i = 1, 2, \dots, N_s$) will contain

- a) a list $\{v_s^i(e_s^i)\}$ of neighbour nodes available from the node v_s^i through arcs $\{e_s^i\}$ incident with this node,
- b) a list of the shortest distances $d_{min}^i(v_s^i, v_s^j)$ between the system node v_s^i and each node v_s^j ($j = 1, 2, \dots, N_s$), measured as the length of the shortest path between v_s^i and v_s^j .

To work out a structural description of a distributed algorithm of the static allocation of a parallel program we will let partition the program graph G_p into N_p nodes and suppose that with each program node v_p^k ($k = 1, 2, \dots, N_p$) is conjugated a decision-making entity consisting of a local environment interpreter E^k and a local decision-making unit U^k (see Fig. 2).

The structural description of the local environment interpreter E^k contains

- a) information about the neighbour relations of a program node v_p^k , i.e. a list $\{\alpha_{r_k}\}$ of its neighbour program nodes, where r_k — a number of neighbours of the node v_p^k ,
- b) a list $(a_{1_k}, a_{2_k}, \dots, a_{r_k})$ of weights of edges incident with the node v_p^k and a weight b_k of the node,

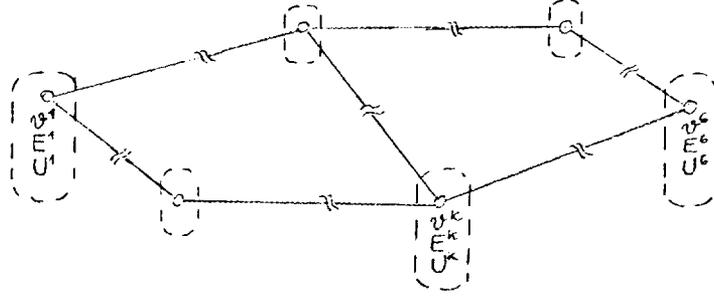


Figure 2: A decision-making entity consisting of a local environment interpreter E^k and a local decision-making unit U^k is conjugated with k - *th* node of a program graph.

- c) an actual location of a given node v_p^k and its neighbours in the system graph, i.e. a list $\{s^i(v_p^k), (s^i(v_p^{\alpha r_k}))\}$ of corresponding system nodes,
- d) a list of actual locations of programs nodes which have visited lately the system node v_p^k where actually is located a given node v_p^k , i.e. a list $\{s_{adr}^i(v_p^k)\}$,
- e) a local function C_i^k describing some actual cost relations in the system graph between given program node v_p^k and its neighbour program nodes $v_p^{\alpha r_k}$ and its previous value C_{old}^k .

A semantics of the distributed algorithm of the static allocation of a program graph is given in Section 4.

3. AUTOMATA GAMES WITH LIMITED INTERACTIONS

To provide a theoretical background for learning automata (for a concept of learning automaton and some its applications, see e.g. [15]) based algorithm of allocation we present below a model of automata games with limited interactions. In the model [18, 16, 17] we suppose that

- a) given a team of automata players $A^1, A^2, \dots, A^k, \dots, A^N$,
- b) for each automaton A^k given a finite set $\{y^k\}$ of its actions,
- c) for each A^k given a payoff function $P^k(y^k, y^{\alpha 1_k}, y^{\alpha 2_k}, \dots, y^{\alpha r_k})$ which depends only on limited number of automata - players: its action y^k and actions of its r_k neighbours in the game ($r_k \ll N$). The meaning of P^k is the expected value of a reward for an automaton A^k for given its action y^k and given actions of its neighbours. It is convenient to represent an interaction in the game by a directed graph where vertices correspond to automata - players, input arcs define players whose strategies influence the payoff function of a player and output arcs define players whose payoff functions depend on a strategy of a given player (see Fig 3),

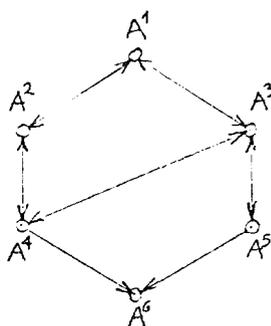


Figure 3: Example of an interaction graph of automata game with limited interactions with a number of players $N = 6$.

- d) the game is played this way that at discrete moment of time $t = 0, 1, 2, \dots$ each player selects independently his own strategy to maximize its own payoff. It is supposed that automata have no a priori information about the game, i.e. about payoff functions, their neighbours or a number of players in the game. They choose their actions only on the base of their single rewards and penalties,
- e) a solution for such a game is the Nash equilibrium point i.e. an N —tuple of actions, one for each player, such that anyone who deviates from it unilaterally cannot possibly improve its payoff.

It is known [16, 17] that automata team is able to find in the dynamic process of the game the Nash equilibrium point. The question which arises here concerns the average value

$$\bar{P}(y^{1*}, y^{2*}, \dots, y^{N*}) = \left(\sum_{k=1}^N P^k(y^{1*}, y^{2*}, \dots, y^{N*}) \right) / N$$

of the payoff received by automata' team in the Nash equilibrium point $(y^{1*}, y^{2*}, \dots, y^{N*})$. Calculating the average automata team payoff for all combinations of automata actions in the game we may find the actions' combination, corresponding to the maximal price point (or points) i.e. the point providing the maximal average payoff received by the automata team. Unfortunately, the maximal price point very often does not correspond to the Nash point and the average payoff received by automata team can be very low.

The solution for the problem in the case of homogeneous automata games with limited interactions (the interaction graph of the game is regular) is introducing into the game a distributed procedure of a conjugate exchange process [18, 20]. The following theorem is the result of introducing the notion of the conjugate exchange process:

Theorem 1 *Introducing the conjugate exchange process into the homogeneous game with limited interactions transforms the maximal price point into the Nash point.*

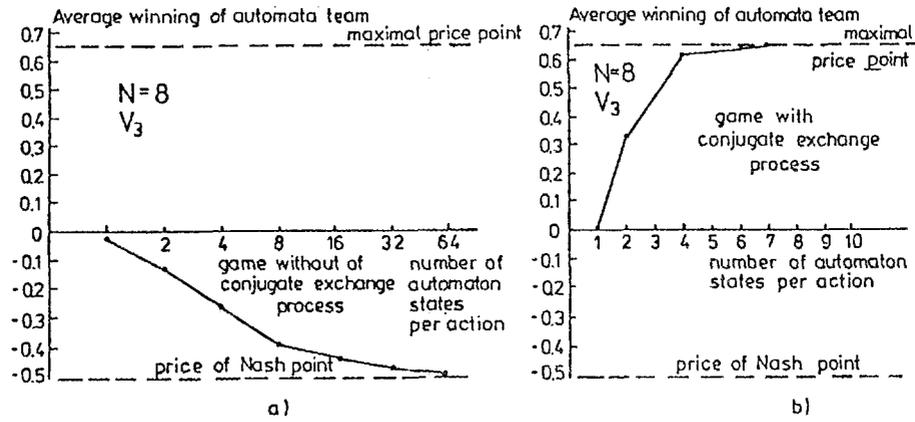


Figure 4: The average winning of the automata team in a homogenous automata game with limited interaction (number of players $N = 8$): (a) game without the conjugate exchange process, (b) game with the conjugate exchange process.

The conjugate exchange process in a game is equivalent an organization of a coalition between neighbours in the game, where each player takes part simultaneously in r coalitions, where r is the degree of the interaction graph.

Fig. 4 shows some results of simulation study of homogeneous automata games with the conjugate exchange process. It can be seen that the team of learning automata is able to find the maximal price point providing for the team the maximal average payoff possible in the game (for more details, see [16]).

4. PARALLEL AND DISTRIBUTED ALGORITHM OF STATIC ALLOCATION

We will consider a process of searching of an optimal static allocation of a program graph in a parallel computer as a dynamic learning automata game with limited interactions [18, 19]. We interpret a program graph (Fig. 1a) as an interaction graph (Fig. 3) of automata' game. We use a learning automaton A^k as a local decision-making unit U^k (Fig. 2) interacting with an environment by a local environment interpreter E^k . We suppose that each automaton A^k ($k = 1, 2, \dots, N_p$) is iniately placed into some system node v_s^i ($i = 1, 2, \dots, N_s$) as a part of a decision-making unit conjugated with a program node v_p^k .

We suppose that in each node v_s^i of the system graph exists a standart description of the type $(e_1^i, e_2^i, \dots, e_{r_k}^i)$ of edges incident with this node. Each automaton A^k will have a set of its $r_k + 1$ actions, i.e. the set $(y_0, y_1, y_2, \dots, y_{r_k})$ which can be interpreted the following way: y_0 —do not move (stop), y_1, y_2, \dots, y_{r_k} —move to a neighbour system node which is available by edges $e_1^i, e_2^i, \dots, e_{r_k}^i$ respectively (Fig. 5).

This way we allow for each automaton to migrate in a system graph together with the decision-making entity and conjugated with it the program node. The aim of each migrating

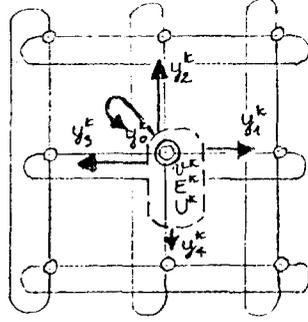


Figure 5: Automaton A^k (with a program node v_p^k and a local environment interpreter E^k) is located in some node of a system graph. Its actions correspond to alternatives to move to a neighbour system node or stay at given location.

automaton is to minimize a local cost function C_i^k of a program node v_p^k located actually in a system node v_s^i .

We suppose that the cost functions C_i^k will be defined as

$$C_i^k = C_{1i}^k + C_{2i}^k,$$

where C_{1i}^k is a heuristics measuring the average communication time between given program node v_p^k located at the system node v_s^i and neighbour program nodes of the node v_p^k , located in some system nodes, and C_{2i}^k is a heuristics responsible for balancing of computational load of program nodes. We suggest to define these functions the following way:

$$C_{1i}^k = 0.5 \sum_{l=1}^{r_k} a_{kl} * d_{min}^i(s^i(v_p^k), s^j(v_p^l)),$$

where: a_{kl} —the time needed to transfer data in the given computer system between neighbour program nodes v_p^k and v_p^l when they are located in the neighbours system nodes; $d_{min}^i(s^i(v_p^k), s^j(v_p^l))$ —the minimal distance between system nodes i and j where are located program nodes v_p^k and v_p^l respectively; r_k —a number of neighbour program nodes of the node v_p^k , and

$$C_{2i}^k = \begin{cases} b_k, & \text{if program node } v_p^k \text{ is located} \\ & \text{in the system node } v_s^i \\ & \text{without nodes being its neighbours} \\ & \text{or nodes having common neighbours with it} \\ b_k + \sum_{n=1}^{n_i} b_n, & \text{if program node } v_p^k \text{ is located in the } v_s^i \\ & \text{together with its neighbour nodes} \\ & \text{or nodes having common neighbours with it,} \end{cases}$$

where: b_k, b_n —running times of program nodes v_p^k and v_p^n respectively, located in the system node v_s^i ; n_i —a number of neighbour program nodes of the v_p^k or nodes having common neighbours with it, located in the system node v_s^i .

We can see that the local function of each automaton depends on its location in the system graph and locations only its neighbour program nodes. A location of each program node is in its turn a function of automaton parameters - automaton actions, which give possibility of migration of the automaton. The automata migration is an adaptive, stochastic and cooperative process of minimizing local cost functions assigned to automata and needs maintaining a communication between given automaton and its neighbours. After each automaton decision concerning its moving the automaton should inform its neighbours about its new position in the next step, and it also should receive messages from its neighbours about their new positions to be able to calculate a new value of its cost function.

In the result of the process of local interactions between automata we can expect achieving by the automata team an equilibrium point characterized by a set of locations of automata in the system graph which will provide for them stable values of their local cost functions. To avoid reaching local minima and provide the possibility to achieve by the automata team a global minimum it is necessary to introduce the exchange process [18, 20] between automata. The conjugate exchange process is a process of exchanging between neighbour automata information about local values of their cost functions C_i^k and calculation a modified cost function \bar{C}_i^k :

$$\bar{C}_i^k = (C_i^k + \sum_{l=1}^{r_k} C_j^l) / (r_k + 1),$$

where C_j^l —a value of the cost function of l -th neighbour of k -th automaton. The global equilibrium point reached by the automata team is connected with minimization by the automata in a distributed manner the following global function:

$$\min \left(\sum_{k=1}^{N_p} \sum_{i \in (1,2,\dots,N_s)} C_i^k \right) / N_p.$$

In the case when weights of edges and nodes of the graph G_p are equal 1, the global function describes the total average distance between the nodes of the graph G_p in the graph G_s . It is equivalent to the mapping problem [21].

The algorithm of a distributed allocation of a program graph into a system graph can be presented now. The algorithm has a sequential part providing a computational model of a distributed environment and parallelly implemented dynamic process of a program graph allocation.

Algorithm

SEQ

1. Partition the program graph G_p into N_p nodes and provide for each k – th node:
 - a) $(\alpha_{1k}, \alpha_{2k}, \dots, \alpha_{r_k})$ (* numbers of neighbour nodes *)
 - b) $(a_{1k}, a_{2k}, \dots, a_{r_k})$ (* weights of edges *)
 - c) b_k (* weight of the node k *)
2. Assign initially (e.g. randomly) each k – th program node with conjugated with it an automaton A^k and a local environment interpreter E^k into i – th system node and define $(s^i(v_p^k), s^{j_1}(v_p^{\alpha_{1k}}), \dots, s^{j_{r_k}}(v_p^{\alpha_{r_k}}))$ (* locations of k – th program node and its neighbours in the system graph *)

PAR

E^k : (* local environment interpreter *)

1. Set: $n_{iter}^k := 0, l_k := 0$

(* l_k - counter of interactions between automata team
for a given configuration of automata locations in the system graph, n_{iter}^k -
counter of iterations (games) of the allocation algorithm *)

A^k : (* ε - automaton *)

2. Choose (e.g. randomly) your current action y^k (i.e. imitate moving to the neighbour system node), send to your r_k neighbours a message concerning the new action and wait for messages concerning their new actions

E^k :

3. Calculate the cost function C_i^k
4. Send the value C_i^k to your r_k neighbours and receive values of their C_j^i
(* exchanging process *)
5. Calculate modified cost function \overline{C}_i^k
6. Store: $C_{old}^k := \overline{C}_i^k$ and $y_{old}^k := y^k$

A^k :

7. Define (randomly) a new action y^k
8. Perform Steps 3, 4 and 5
9. Accept as y_{new}^k :

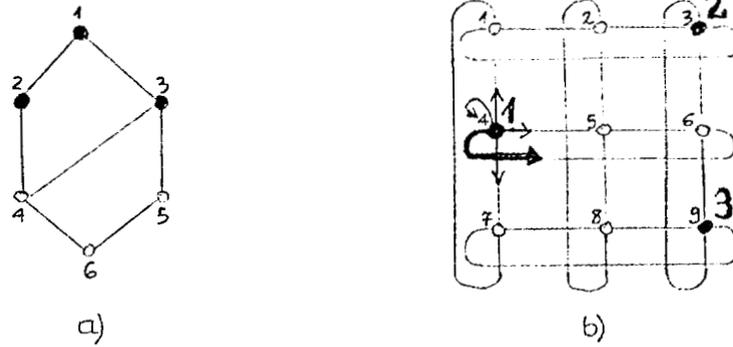


Figure 6: Initial allocation of the first three program nodes in the system graph.

y^k , with probability $1 - \varepsilon$ ($0 < \varepsilon < 1$), if $\bar{C}_i^k \geq C_{old}^k$ or y_{old}^k , if $C_{old}^k > \bar{C}_i^k$
and with probability $\varepsilon/(r_k + 1)$ any of $r_k + 1$ actions of the automaton

10. Store: $C_{old}^k := \bar{C}_i^k$, $y_{old}^k := y^k$, $y^k := y_{new}^k$

11. $l_k := l_k + 1$; if $l_k < L$ then goto Step 8 (* L - a number of interactions *)

12. $l_k := 0$

E^k :

13. Move to neighbour system node corresponding to current action y^k , send your new address to your r_k neighbours and wait for messages concerning their new locations

14. Set: $n_{iter}^k := n_{iter}^k + 1$

15. If $n_{iter}^k < T$ then goto Step 8

(* T - a number of iterations (games) of the allocation algorithm *)

5. AUTOMATA BASED ALLOCATION ALGORITHM - A SIMULATION BY HAND

To provide a better insight into proposed allocation algorithm we discuss here the most essential steps of the algorithm. For this reason we suppose that Steps 1 and 2 ((SEQ) were performed and nodes of the program graph from Fig. 1a were randomly located into the system graph from Fig. 1b. Fig. 6 shows possible initial allocation of the first three program nodes in the system graph, i.e. the program node 1 is located in the system node 4, the program node 2 is located in the system node 3 and so on.

Let us have a look at the steps of the algorithm from point of view of learning automaton (we use a stochastic ε - automaton [18-19] suitable to operate in a deterministic environment) conjugated with e.g. the program node 1. The automaton A^1 has the set of

actions $\{4, 1, 5, 6, 7\}$ with the following meaning: 4 - do not move, 1 - move to the system node 1 and so on. According to Step 2 (PAR) the automaton A^1 chooses randomly its action, e.g. $y^1 = 7$ (i.e. imitate moving to the system node 7) and informs neighbours automata A^2 and A^3 about its current action, waiting at the same system node for messages concerning their current actions. To simplify our discussion we assume that weights of nodes and edges of the program graph are equal 1 and actions of neighbours automata A^2 and A^3 are equal $y^2 = 3$, $y^3 = 9$ (i.e., do not move) respectively and they do not change them in time.

After receiving messages random environment interpreter E^1 can calculate the cost function C_4^1 (Step 3). It evaluates first a minimal distance (a number of single hops) to its neighbours. The minimal distance is evaluated between the system node pointed by the action of the automaton A^1 (node 7) and nodes pointed by the automata A^2 and A^3 (nodes 2 and 9 respectively). These values are equal 2 and 1 respectively and a value of the cost function is equal $C_4^1 = 1.5$ (see the last column of Table 1).

Table 1. Minimal Distances of the Program Node 1 to Its Neighbours Depending on an Action of the Automaton A^1 and Its Current Value of the Cost Function

actions of automaton A^1	4	1	5	6	7
$d_{min}^1(v_p^1(y^1), v_p^2(y^2))$	2	1	2	1	2
$d_{min}^1(v_p^1(y^1), v_p^3(y^2))$	2	2	2	1	1
C_4^1	2	1.5	2	1	1.5

In a similar way the automaton evaluates its next action (Steps 7, 8) to find the best one in Step 9. In Step 11 the algorithm returns control to Step 8 and the sequence of Steps 8 - 10 is again repeated, eventually L times. During L iterations automata do not change their locations but only simulate changing and evaluate their best actions. As can be seen from Table 1 the action 6 minimizing local function is the best for the automaton A^1 and one can expect that the automaton will finally move to the system node 6 at Step 13 of the algorithm.

Real situation defined by the algorithm (Steps 2 - 11) is however more complex because the algorithm allows all automata to change their actions at the same step (Step 10). Fortunately, this sequence of steps exactly corresponds to the model of learning automata team playing the game with limited interactions (see, Section 3), with the payment function P corresponding to given allocation of the program nodes in the system graph. Initial allocation of the program graph nodes in the system graph defines the payment function P^1 of the game. The game is played L times and at the end of the game automata actions point a new plan of allocation minimizing the average value of computer and communication delays between program nodes for given stage. The new plan of allocation is performed in Step 13 where a new game with a payment function P^2 is established. This way the allocation algorithm may be considered as a sequence of T games with payment functions P^1, P^2, \dots, P^T , each improving a plan of allocation.

6. CONCLUSIONS

The problem of static allocation of a program graph in a parallel computer was considered. Parallel and distributed algorithm of the static allocation problem was presented. The allocation process was interpreted as a sequence of dynamic games of learning automata. Some results concerning a global behaviour of the automata team achieved only by a local cooperation of automata taking part in games with limited interactions were applied. It is expected that the algorithm will be able to produce a suboptimal or optimal allocation corresponding to the minimum of the average total value of locally defined cost functions.

References

- [1] C. C. Price and S. Krishnaprasad, "Software Allocation Models for Distributed Computing Systems", Proc. 4th. Int Conf. Dist. Comp. Systems, May 1984.
- [2] F. Berman, "On Mapping Parallel Algorithms into Parallel Architectures",
- [3] P. Y. R. Ma, E. Y. S. Lee and J. Tsuchiya, "A Task Allocation Model for Distrib. Comp. Syst.", *IEEE Trans. on Computers* **31**(1) (1982).
- [4] C. P. Kruskal and A. Weiss, "Allocating Independent Subtasks Parallel Processors Extended Abstract", Int. Conf. Parallel Proc., 1984.
- [5] J. L. Gaudiot and J. I. Pi, "Program Graph Allocation in Distributed Multicomputers", *Parallel Comput.* **7** (1989).
- [6] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessors", *J. of Parallel and Distributed Computing* **7** (1989).
- [7] F. C. H. Link and R. M. Keller, "The Gradient Model Load Balancing Method", *IEEE Trans. Software Eng.* **13**(1) (1987).
- [8] L. Davis, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufman Publishers, Los Altos, California (1990).
- [9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publ. Comp., Inc. (1989).
- [10] Z. Michalewicz, J. Krawczyk, M. Kazemi and C. Janikow, "Genetic Algorithms and Optimal Control Problems", Proc. of the 29th IEEE Conf. on Decision and Contr., Honolulu, Dec. 5-7, 1990.
- [11] E. H. L. Aarts and J. H. M. Korst, "Boltzmann Machines for Travelling Salesman Problems", *European Journal of Operational Research* **39** (1989). *J. of Parallel and Distributed Computing* **4** (1987).
- [12] L. Xu and E. Oja, "Improved Simulated Annealing, Boltzmann Machine and Attributed Graph Matching", Proc. of the Neural Networks :EURASIP Workshop 1990, Sesimbra, Portugal 1990

- [13] S.Lakshmiarhan, *Learning Algorithms Theory and Applications*, Springer-Verlag, New York Heidelberg Berlin, (1981).
- [14] K. S. Narendra and M. A. L. Thathacher, "Learning Automata - a Survey", *IEEE Trans. Syst., Man, Cybern.*, **4** (1974).
- [15] B. J. Oomen and D. C. Y. Ma, "Deterministic Learning Automata Solutions to the Equipartitioning Problem" , *IEEE Trans. on Computers* (**37**)(1) (1988).
- [16] M. L. Tsetlin, *Automaton Theory and Modelling of Biological Systems*, New York: Academic, (1973)
- [17] V. I. Varshavskii, *Collective Behaviour of Automata*, Moscow, Nauka, (in Russian) (1973)
- [18] F.Seredynski, "Self-organization in Homogenous Networks of Learning Automata", to appear in Proceedings of the 6th Czechoslovak Annual AI Conference, Prague, June 25-27, 1991.
- [19] F.Seredynski , "Allocation of a Program Graph in Parallel Computers by Learning Automata", Proc. of the 5-th Int. Workshop on Parall. Proc. by Cellular Automata and Arrays PARCELA '90, Berlin, Germany, Sept. 17-21, 1990.
- [20] V. I. Varshavskii, A. M. Zabolotny and F. Seredynski, "Homogenous Games with a Conjugate Exchange Process", *Izv. Acad. of Science USSR Tech. Kibern.* (**6**), in Russian) (1977).
- [21] S. H. Bokhari, "On the Mapping Problem", *IEEE Trans. on Computers* **30**(3) (1981).

NOTES ON ROUGH INFERENCE

Brian Shay

Department of Mathematics and Statistics
Hunter College, CUNY
695 Park Avenue
New York, New York 10021

ABSTRACT

I propose a general strategy for defining theories of non-monotonic inference, and a particular theory for use in connection with methods for modeling uncertain belief and reasonable action. Although non-monotonic inference is defined without reference to subjective probability, my methods for modeling uncertain belief and reasonable action are based on subjective probability. I can guarantee that the lottery paradox will not arise in application of the particular theory in connection with these methods.

1. INTRODUCTION

These notes are intended as a brief introduction to interlocking mathematical formalisms for non-monotonic inference, uncertain belief and reasonable action. I have presented these formalisms elsewhere with substantial mathematical development. Here, the treatment will be informal and interpretive. The comprehensive model is as well suited for machine reasoning as for reasoning about everyday matters: that is, there are no essential gaps in the mathematical development.

There are few influences on this project from the AI literature, particularly the literature on non-monotonic reasoning. The model for uncertain belief is related to a model of Fagin and Halpern, but I worked independently, even reporting results a few months earlier (see Shay [1], Fagin/Halpern [2]). The independent development should not be surprising, since it proceeded from classical (inner and outer) measure theory, whose rudiments are familiar to every mathematician.

There are substantial similarities to Shafer's theory of evidence in the method for modeling uncertain belief. My explicit intention was to develop a formalism that could represent Shafer's models, but to supplant Dempster's rule with an alternate method for combining models, namely, nondeterministic probability extensions. Necessary and sufficient conditions for such combination and many technical details are to be found in [1]. Fagin and Halpern had other goals, but observed also that classical measure theory was adequate to

represent belief function models. This result was surprising, as there had been much debate about whether or not classical probability theory was adequate to cover the concerns of Shafer and other adherents to his approach.

The method for modeling reasonable action is patterned, to an extent, on the notion of random variable, and is a natural accretion to the method for modeling uncertain belief. It is intended as a substitute for utility-based decision theory. I have seen no similar model in the literature.

Neither have I seen any close counterpart to the strategy for defining theories of non-monotonic reasoning, or the particular theory that I have defined for the comprehensive model. In particular, unlike many approaches to non-monotonic reasoning, my approach is not intrinsically "probabilistic", though there is an excellent fit with subjective probabilistic ideas. Moreover, my approach, in combination with probabilistic ideas, can readily circumvent the lottery paradox. This must be counted as unusual (see e.g. Kyburg [3]). A similar claim has been made by Bacchus [4] for his own approach. However, in [4] he has far more limited goals in his approach to non-monotonic inference, limiting its application to default reasoning concerning "statistical" probabilities. I don't know of any other such claim, but I have little access to work in progress.

The present account might appear at first to be over-burdened with discussion, at the expense of mathematical development. I am limited by space, but I also regard the present emphasis as suitable. There has been substantial confusion concerning Dempster-Shafer formalism, for example, focused on multiple interpretations of "evidence", "belief", "updating", etc. Nevertheless, the mathematical formalism that underlies much quarreling is utterly simple. Recent examples of interpretive essays, intended to clarify these issues, are of Dubois and Prade [5], and Pearl [6]. The inner and outer measure approach to belief representation that I offer here is superficially a special case of Shafer's belief function formalism. However, it can be demonstrated as well that the belief function formalism is a special case in measure theory. But these embeddings are not inverses. This subtlety indicates at least the need for clear distinctions, including distinct terminology. Ample discussion is otherwise intended to prevent the quarreling that has erupted in the Dempster-Shafer arena from spilling over.

Ideas for this project came to me as a result of focusing on Aristotle's "Art of Rhetoric" [7] rather than on current literature in AI. Aristotle is concerned in part with specifying what is rhetorical syllogism ("enthymeme"), wrought from probabilities and signs. My concern, in the comprehensive model, is to represent accumulation of uncertain belief in considering arguments that might be defeasible, and a relationship between beliefs and reasonable action. There is a natural link between these concerns and themes

of classical rhetoric (that has not greatly influenced AI methodology, I think). Although action can be based on defeasible arguments, there are no regrettable actions in my approach: any action taken as a result of accepting a defeasible argument might have been taken reasonably without such acceptance. This resilience is surprising (to me).

2. BELIEF

I choose subjective probabilities for measuring beliefs. A belief is an idea that is believed to an extent between 0 and 1. This is conventional, but I do not require that every idea be believed. A Boolean algebra of ideas under consideration (an agenda) contains a sub-algebra of beliefs. A probability function measures extents of belief in beliefs.

The ideas can serve as an agenda for more than one actor. However, the beliefs of one actor might differ from the beliefs of another (different subalgebras), and should an idea be held as a belief by each of two actors, there needn't be agreement as to the extents of belief (different probability measures on the intersection of the two subalgebras).

Truth and falsity are not concerns of this project. The units of Boolean algebras will be CERTAINTY and NONSENSE.

Inference is based primarily on the structure of the Boolean algebras (material implication). However, I assume that actors judgments are summarized in part by compatibility relations between beliefs and ideas. If a belief is not compatible with the negation of an idea (in the judgment of an actor) then that belief is a sure sign (for him) of that idea. For one actor, "He is a Republican" is a sure sign that "He is hard-hearted"; for another actor, "He is a Republican" is compatible with "He is hard-hearted" and with "He is not hard-hearted". The notion of sure-sign is intended to represent Aristotle's notion of necessary sign. Compatibility is intended to represent Aristotle's notion of sign. (There is a natural definition of to be a sign to an extent, but it is an outcome, rather than presupposition, of the theory). Compatibility relations satisfy axioms that guarantee that sure-significance is a (partial) representation of material implication in quotient Boolean algebras. Thus, different actors appear to be reasoning about the same propositions (elements in a common Boolean algebra), yet exhibit a form of variation that is common in everyday speech (and even in formal deliberations) that can be interpreted as reasoning from distinct (quotient) Boolean algebras. An example of an axiom is: a belief is compatible with another if and only if their conjunction is believed to extent greater than 0. Another is: a belief is compatible with the disjunction of two ideas if and only if it is compatible with one of them. Naturally implication is a special case of sure-significance, if the antecedent is a belief.

Sure-significance is associated with inference, not rough inference, that is reasoning without paying heed to every special case. The actor who believes Republicans are hard-hearted is not prepared to consider exceptions. It is, to him, nonsensical that there are exceptions. More clear-cut instances can be found in scientific discourse. One generation of scientists will adopt principles that will be overturned by the next generation (e.g. "ontogeny recapitulates phylogeny"). Until new discoveries, such principles are used as "laws of nature", intended to cover all special cases.

3. CUMULATING BELIEF

To come to believe an idea that is not a belief is to choose an extent of belief for that idea. The choice is constrained, for the laws of subjective probability will apply to the extended set of beliefs. The constraints are these: i) if a present belief is a sure sign of an idea, then the extent of belief in that present belief is a lower bound for the feasible extents of belief for that idea; ii) if an extent of belief is feasible for an idea, 1 minus that extent of belief is feasible for the negation of that idea.

Together, these constraints define for each idea an interval of feasible extents of belief. If an idea is a belief, that interval is degenerate, a singleton. To come to believe an idea that is not a belief is to choose a feasible extent of belief from the interval associated with it.

Evidently, the lower and upper bounds are closely associated with the belief and plausibility functions of Dempster-Shafer theory. They are inner and outer measures of classical measure theory. However, belief and plausibility functions are not generally inner and outer measures, yet canonical inner and outer measures can be associated with a belief or plausibility function. Consequently, there is certain to be confusion if I rely on the the Dempster-Shafer terminology.

I say that the lower bound of feasible extents of belief is the assurance of an idea, and the upper bound of feasible extents of belief is the promise of an idea. Evidently, promise is the upper limit of assurance that develops as other ideas are adopted first as beliefs.

Non-trivial theorems concerning extensions of probability measures are required to support these comments. They are proved in [1].

Evidently, there is a sense in which this approach supports intervalistic probability models. However, I do not identify an interval with extent of belief. A belief function in Dempster-Shafer theory determines a plausibility function (see Shafer [8]) and, consequently, this identification has been effectively adopted there.

4. RULES GOVERNING REASONABLE ACTION

In [9] I link actions to an agenda by rules of the sorts:

X will do A if he believes or comes to believe B
to extent lying in interval I;

X will do A only if he believes or comes to believe B
to extent lying in interval I;

and simple generalizations.

Actions, too, are elements of algebras, and consistency of actions is defined in a manner analogous to compatibility of beliefs and ideas. Rules can trigger inconsistent action. A set of rules that cannot trigger inconsistent action is a consistent plan of action.

Suppose B is not believed to any extent, but its feasible extents of belief lie within interval I. Should X undertake to do A? The sure-thing principle suggests that he should, but there will be differences of opinion on this point.

Since an actor's regula concerning an agenda can refer to ideas not believed, he can be motivated to choose a feasible extent of belief just so that he is thereby regulated to undertake an action, or permitted to undertake an action. This is realistic, and a welcome feature.

5. THEORIES OF ROUGH INFERENCE

I consider a non-monotonic inference operator, $.ri.$, to be a relation on a Boolean algebra: $P.ri.Q$ signifies that P can be inferred, roughly speaking, from Q.

I offer a particularly simple strategy for determining whether or not a relation is to be called a rough inference operator:

from a characterization of material implication by axioms,
including consequence monotonicity,
remove consequence monotonicity.

The residual axioms, a theory for rough inference, either characterize material implication or fall short of it barely, failing to guarantee consequence monotonicity. There are numerous non-monotonic inference theories of this sort. A relation in a Boolean algebra satisfying the axioms for a theory for rough inference is a rough inference operator. I do not require that the relation be the largest (perfect) relation satisfying those axioms. Techniques of Touretzky [10] [11] can be applied to edit and extend rough inference operators towards perfection.

As simple and appropriate as this strategy is, I have not seen it proposed elsewhere. Whether an author proposes axioms for a weak inference relation or inference or deduction rules, almost always the cartesian product relation will satisfy the restrictions. An example is the propositional logical system, C, of Kraus et al. [12]. There seems to be a scholarly penchant for ensuring that monotonicity can't be deduced, rather than ensuring that inference is characterized if monotonicity is an additional requirement.

It should be noted that my strategy has no intrinsic connection with probabilistic ideas.

The following axioms are a theory of rough inference:

REFLEXIVITY: each proposition is a rough antecedent of itself;

SEPARATION: a proposition and its negation have no rough antecedents in common other than NONSENSE;

ANTECEDENCE MONOTONICITY: a rough antecedent of a proposition is a rough antecedent of every consequence of that proposition.

For contrast, I will state the axiom of consequence monotonicity. If this axiom is joined to any theory of rough inference, in particular the theory consisting of the three axioms above, than any operator in a Boolean algebra satisfying the axioms will be a sub-relation of material implication.

CONSEQUENCE MONOTONICITY: every antecedent of a rough antecedent of a proposition is a rough antecedent of that proposition.

To illustrate: If a man works hard then, roughly speaking, he will succeed in life; if a man is a prisoner in a state penitentiary, then he works hard. Conclude from consequence monotonicity that: if a man is a prisoner in a state penitentiary then, roughly speaking, he will succeed in life. This is clumsy. The focus of non-monotonic reasoning is to preclude such accounts.

On the other hand: If a man speaks French then, roughly speaking, he will be able to read signs written in French; if a man can read signs written in French, then he will not be easily lost in Paris. According to antecedence monotonicity: if a man speaks French then, roughly speaking, he will not be easily lost in Paris. This account is more natural. The occasional blind speaker of French is an exception, but not a troublesome exception.

The axioms of reflexivity, separation, and antecedence monotonicity seem to be a reasonable core for more specialized theories.

I will add several axioms to specify a theory that can be integrated smoothly with the inner and outer measure approach to representing belief. However, it will be helpful to consider first how rough inference might be used to advance overall project goals.

Belief formation is incremental; cumulation of belief leads to more complex activity. Inference, associated with sure-significance, is the determinant of intervals of feasible extents of belief, and thereby, a partial determinant of admissible action.

Rough inference might then be used to specify sets of feasible extents of belief, subsets of the feasible sets specified by inference, for rough inference dominates inference. How might such feasible subsets be used? A conservative use would be this: an actor might modify his plan of action by application of the sure-thing principle and rules governing action, but substituting feasible sets identified by rough inference for feasible sets identified by inference. The emphasis of this approach is on how to modify plans of action, not how to modify belief. But this is a traditional goal of AI: to learn how to act without regard for troublesome special cases.

An illustration will help clarify this point. Let PW be the proposition: a person has been wounded as victim of a street crime, but there is no longer any present danger. Let SF be the proposition: I hear a shot fired, a scream, and the sound of someone running away. SF is compatible with PW, but is not a sure sign of PW, in my judgment, but I judge SF to be a rough sure sign of PW. Suppose I have a rule: if I come to believe PW to extent between .9 and 1.0, then I will seek out the victim and try to be helpful. On an occasion, I believe SF to extent 1 (I hear, and trust my senses). I act, according to my rule, not because I adopt a belief to extent 1 in PW, but because I have a meta-rule: I will undertake those actions that would be triggered were rough inference to replace inference in specifying feasible extents of belief. This meta-rule is reasonable, as rules are the basis of a plan that is developed before beliefs concerning an agenda are fully specified, and there should be some flexibility with regard to their applicability.

Note that I do not adopt the belief to any extent that PW, but only act as if I had. If I find the "victim" to be frightened, but unhurt, I will not regret the action, as I was free to believe in PW to extent 1, even without application of the meta-rule, and consequently the action I took was a reasonable action, according to my plan, at the time.

Technical issues associated with the feasible sets specified by rough inference are these: i) might it happen that such feasible sets be intervals? ii) might it happen that such feasible sets be non-empty? iii) might it happen that whenever the feasible sets associated with two propositions are both $[0,0]$, that the feasible set associated with their disjunction be also $[0,0]$? (i.e. might it happen that the lottery paradox cannot arise?)

I add axioms in two stages: two axioms will ensure that, in my comprehensive model, feasible subsets determined by rough inference will be sub-intervals of feasible extents of belief determined by inference. A final axiom will guarantee that the lottery paradox will not arise.

STRONG SEPARATION: if one proposition and a second are rough antecedents of a third and of the negation of the third respectively, then the first is a rough antecedent of the negation of the second.

DISJUNCTIVE CLOSURE: if one proposition and a second are rough antecedents of a third, then the disjunction of the first and second is a rough antecedent of the third.

With a few technical niceties (see Shay [13]) the rough inference theory specified by the axioms of reflexivity, separation, antecedence monotonicity, strong separation, and disjunctive closure will specify non-empty subintervals of feasible extents of belief determined by rough inference rather than inference. For every idea, [rough-assurance, rough-promise] is a non-empty subinterval of [assurance, promise]. For use in connection with the comprehensive model, I modify these axioms slightly, taking into account that only beliefs are to be rough antecedents, and that "proposition" can be interpreted as an element of a quotient algebra, specified by an (any) extension of the probability measure representing belief that is consistent with sure-significance.

This theory seems adequate to represent belief formation based on "rhetorical syllogism". As belief formation proceeds "one belief at a time", all that is needed for guidance is restriction of the intervals of feasible extents of belief without degeneration. If the lottery paradox arises, it is resolved in the following manner: an actor chooses convincing but defeasible arguments to support adopting new beliefs, whereupon he revises his judgments concerning rough antecedence non-monotonically. Since judgments of rough antecedence are defeasible in principle, this is not unreasonable progress. Thus, although the lottery paradox can arise, it is not evidence of inconsistent belief.

With the addition of another axiom, the lottery paradox can be avoided altogether.

STRONG CONJUNCTIVE CLOSURE: if one proposition is a rough antecedent of a second, and a third is a rough antecedent of a fourth, then the conjunction of the first and the third is a rough antecedent of the second and the fourth.

Let the rough inference theory specified by the axioms of reflexivity, separation, antecedence monotonicity, strong separation,

disjunctive closure, and strong disjunctive closure be called the theory of almost-sure-significance. In a Boolean algebra with a rough inference relation satisfying these axioms, if a proposition is a rough antecedent of a second, then the first is said to be an almost sure sign of the second.

If the corresponding rough-assurance and rough-promise functions are denoted ra and rp , then, for propositions A and B of the Boolean algebra,

$$ra(A) + ra(B) \leq ra(A \text{ and } B) + ra(A \text{ or } B);$$

$$rp(A) + rp(B) \geq rp(A \text{ and } B) + rp(A \text{ or } B).$$

These are familiar properties of belief and plausibility functions in Dempster-Shafer theory (see [8]).

It follows immediately that the lottery paradox cannot arise: i.e. the upper bound of roughly feasible extents of belief of the disjunction of propositions must be 0 if the corresponding upper bound for each disjunctive component is 0.

The most conservative use of this theory was introduced earlier: to adapt a reasonable plan of action by acting as if the feasible extents of belief were specified by almost-sure-significance, rather than sure-significance, carrying out only those actions triggered by the sure-thing principle, while abstaining from any actual extension of belief. This is a different use of rough inference from guiding belief formation.

It is unreasonable to believe that my proposal for defining theories of rough inference can capture the immense variety of forms of commonsense reasoning. Nevertheless, I have been able to combine simple principles to model rigorously and in a realistic manner two styles of commonsense reasoning, and it is not unreasonable to expect more from this strategy.

6. THE LOTTERY

I offer a more complex illustration:

The following ideas generate a Boolean algebra:

Facts (whose negations are not compatible with any ideas):

A lottery is in progress; there is exactly one winning ticket; a person who has no ticket will not win the lottery.

Other ideas: I have a ticket; I have a winning ticket; there are many tickets; I will win; the lottery is fair; my friend has no ticket; my friend will win; I would be better off without a ticket.

There is a community of bettors attentive to the present lottery. They share the facts about the lottery, at least.

I have these beliefs: I believe facts to extent 1. I believe to extent 1 that I have exactly one ticket, that my friend has no ticket, that my friend will lose. I believe to extent .99 that the lottery is fair, to extent .90 that there are many tickets.

I do not believe to any extent that I will win; nor do I believe to any extent that I will win. These are ideas that I consider believing, but do not believe.

Other bettors will have different extents of belief concerning the fairness of the lottery, for example. I judge that unfairness in the lottery is incompatible with my winning. (If it is unfair, and I am not a conspirator, then I consider it nonsensical that I be the beneficiary). I think, therefore, that the lottery is not fair is a sure sign that I will lose. Others might not share this judgment.

The assurance my beliefs give me for the idea that I will lose is .01; the promise of the idea that I will lose is 1.0, as there are no sure signs (other than NONSENSE) that I will win.

I next consider rough antecedents: I judge that my having but one ticket is, roughly speaking, a sure sign that I will lose. I posit no other rough antecedents that are not sure signs. In particular, I do not judge that my having but one ticket is, roughly speaking, a sure sign that I am better off not having bought a ticket.

The rough-assurance my beliefs give me for the idea that I will lose is 1.0. The rough-promise is 1.0 as well.

I have a rule: that if I come to believe to extent greater than .95 that I will lose, then I will put the lottery out of mind. (That is not to say I would be unresponsive if I were declared the winner). Using the feasible extents of belief associated with almost-sure-significance, I apply the sure-thing principle, and put the lottery out of mind. I do not believe to extent 1 that I will lose, but I act as if I did.

7. CLOSING REMARKS

The example presented in the previous section appeared in [13], to illustrate "rhetorical syllogism". The results on the lottery paradox, and the theory of almost-sure-significance are reported for the first time here. [13] will be revised to follow this account more closely. It should be clear that the frame problem is a natural source of examples of a similar style. Actors can act as if changes of state of record are the changes of state in fact, without believing to extent 1 that they are.

REFERENCES

1. B. Shay, "A Mathematical Theory of Evidence", Hunter College, CUNY (1987).
2. R. Fagin and J. Y. Halpern, "Uncertainty, Belief and Probability", IBM Research Report RJ 6191, IBM (1988).
3. H. Kyburg, "Probabilistic Inference and Non-monotonic Inference", *Uncertainty in Artificial Intelligence 4*, R. D. Schacter, T. S. Levitt, L. N. Kanal, J. F. Lemmer eds., Elsevier, New York (1990).
4. F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge*, MIT Press, Cambridge (1990).
5. D. Dubois and H. Prade, "Evidence, Knowledge and Belief Functions", submitted to *International Journal of Approximate Reasoning*.
6. J. Pearl, "Reasoning with Belief Functions: an Analysis of Compatibility", *International Journal of Approximate Reasoning* 4(5) (1991).
7. Aristotle, *The "Art" of Rhetoric*, J. H. Freese tr., Loeb Classical Library, Harvard University Press, Cambridge (1926).
8. G. A. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press (1979).
9. B. Shay, "A Sure Thing Principle", Hunter College, CUNY (1990).
10. D. S. Touretzky, *The Mathematics of Inheritance*, Pitman, London (1986).
11. D. S. Touretzky and R. H. Thomason, "Non-monotonic Inheritance and Generic Reflexives", Proceedings of the Seventh National Conference on Artificial Intelligence, AIJL '88, pp. 433-438.
12. K. Kraus, D. Lehmann and M. Magidor, "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics", *Artificial Intelligence* 44 (1990).
13. B. Shay, "Rough Inference and Rhetorical Syllogism", Hunter College, CUNY (1991).

BOTTOM-UP EVALUATION IN INDEFINITE DEDUCTIVE DATABASES

Rajshekhar Sunderraman

Computer Science Department
The Wichita State University
Wichita, Kansas 67208

Abstract

A bottom-up approach to evaluate non-Horn rules in indefinite databases is presented. Tabular structures, called *C-tables*, are used to represent disjunctive facts. Algebraic operations on *C-tables* are used to evaluate non-Horn rules. The bottom-up approach computes the fixpoint semantics of disjunctive deductive databases.

1 INTRODUCTION

In recent years, the field of deductive databases has been the focus of intense research and there has been a dramatic advance in the understanding of the theoretical and practical issues involved. A substantial amount of effort has gone into definite deductive databases, a subclass of deductive databases in which only Horn clauses are allowed. The semantics of such databases are fully understood and there has been a great deal of research dealing with implementation issues, particularly in query optimization in the presence of recursive rules. This research has culminated in various experimental systems like NAIL! [MUG86], LDL [NT88], and Postgres [SR86], the utility of which have been successfully demonstrated. Therefore, it is not unreasonable to assume that within the next decade, commercial systems with deductive capabilities will become available.

In the presence of a large number of facts and relatively few rules, as is the case with definite deductive databases, the bottom-up evaluation of rules (with optimization techniques like magic sets) performs much more efficiently than top-down evaluation. Moreover, the bottom-up evaluation using the relational algebra can take advantage of the efficient database access techniques involving joins that are a part of modern day relational database management systems. For these and other reasons the successful experimental systems like LDL and NAIL! have opted for the bottom-up evaluation model.

Indefinite deductive databases, which allow non-Horn clauses, are a subject of study by many researchers. Many of the semantic issues for indefinite deductive databases have recently been solved. The declarative, fixpoint and procedural semantics for disjunctive logic programs have been presented in [MR90]. Since deductive databases and logic programs share the same form of representation (clausal form), most of the semantics for disjunctive logic programs can

be carried over to indefinite deductive databases. In particular, if we disallow function symbols and restrict ourselves to pure non-Horn clauses (atoms on both sides of the \leftarrow symbol), the semantics of indefinite deductive databases is very well understood. The declarative semantics corresponds to the set of minimal models of the database and the fixpoint semantics is obtained using the T_P^I operator. The equivalence of declarative and fixpoint semantics is shown in [MR90]. These semantics will be the basis of our bottom-up approach to evaluate non-Horn rules.

In this paper, we present a bottom-up algebraic approach to evaluate non-Horn clauses. We use a modified version C-tables of [IJ81] to represent the extensional database of disjunctive facts. The algebraic operations defined on C-tables are then used to evaluate the non-Horn clauses of the intensional database.

2 BACKGROUND

As far as this paper is concerned, we shall restrict indefinite deductive databases to consist of non-Horn clauses of the form

$$A_1, \dots, A_n \leftarrow B_1, \dots, B_m$$

where A_i s and B_j s are atomic formulas that do not contain function symbols.

2.1 Semantics of Indefinite Deductive Databases

The *declarative* semantics of indefinite deductive databases is based on Herbrand models. Such databases do not possess a unique smallest Herbrand model, but instead have a collection of minimal Herbrand models [Min82]. The following theorem illustrates the declarative semantics:

Theorem 2.1 ([Min82]) *For an indefinite deductive database P and for every positive clause E , $P \models E$ if and only if E is true in every minimal model of P .*

The *fixpoint* semantics is based on the immediate consequence operator T_P^I defined in [MR90]. For this, we need the notion of the *extended Herbrand Base*, EHB_P for database P , which is defined to be the set of all finite disjunctions of different atoms of the Herbrand Base HB_P . T_P^I is defined as follows:

$$T_P^I(S) = \{C \in EHB_P \mid \begin{array}{l} C' \leftarrow B_1, \dots, B_n \text{ is a ground instance of a program} \\ \text{clause in } P \text{ and } B_1 \vee C_1, \dots, B_n \vee C_n \text{ are in } S \text{ and} \\ C'' = C' \vee C_1 \vee \dots \vee C_n, \text{ where } \forall i, 1 \leq i \leq n, C_i \text{ can be null, and} \\ C \text{ is the smallest factor of } C'' \}. \end{array}$$

Define the powers of T_P^I as follows:

$$\begin{aligned} T_P^I \uparrow 0 &= \emptyset \\ T_P^I \uparrow (i+1) &= T_P^I(T_P^I \uparrow (i)) \\ T_P^I \uparrow \omega &= \text{lub}\{T_P^I \uparrow (i) \mid i < \omega\} \end{aligned}$$

Example 2.1 Consider the following database taken from [GM89]:

$$\begin{aligned}
r_3(a_1, a_3) \vee r_3(a_3, a_4) &\leftarrow \\
r_3(a_4, a_6) &\leftarrow \\
r_4(x, y) &\leftarrow r_3(x, y) \\
r_4(x, y) &\leftarrow r_3(x, z), r_4(z, y) \\
r_1(x), r_2(y) &\leftarrow r_4(x, y) \\
r_1(y) &\leftarrow r_3(x, y)
\end{aligned}$$

$T_P^I \uparrow \omega$ for this database is:

$$\begin{aligned}
T_P^I \uparrow \omega = & \{r_3(a_1, a_3) \vee r_3(a_3, a_4), r_3(a_4, a_6), \\
& r_1(a_6), r_1(a_3) \vee r_3(a_3, a_4), r_1(a_4) \vee r_3(a_1, a_3), \\
& r_3(a_1, a_3) \vee r_4(a_3, a_4), r_3(a_3, a_4) \vee r_4(a_1, a_3), r_4(a_4, a_6) \\
& r_1(a_1) \vee r_2(a_3) \vee r_3(a_3, a_4), r_1(a_3) \vee r_1(a_4), r_1(a_3) \vee r_2(a_4) \vee r_3(a_1, a_3), \\
& r_1(a_3) \vee r_3(a_3, a_4), r_1(a_3) \vee r_4(a_3, a_6), r_1(a_4) \vee r_2(a_6), \\
& r_1(a_4) \vee r_4(a_1, a_3), r_3(a_1, a_3) \vee r_4(a_3, a_6), r_4(a_1, a_3) \vee r_4(a_3, a_4), \\
& r_4(a_1, a_3) \vee r_4(a_3, a_6), \\
& r_1(a_1) \vee r_1(a_4) \vee r_2(a_3), r_1(a_1) \vee r_2(a_3) \vee r_4(a_3, a_4), \\
& r_1(a_1) \vee r_2(a_3) \vee r_4(a_3, a_6), r_1(a_3) \vee r_2(a_4), \\
& r_1(a_3) \vee r_2(a_6)\}
\end{aligned}$$

We can remove the disjunct $r_1(a_3) \vee r_2(a_4) \vee r_3(a_1, a_3)$ since it is subsumed by the disjunct $r_1(a_3) \vee r_2(a_4)$. \square

We shall return to this example at a later point in the paper.

The following theorem illustrates the fixpoint semantics of indefinite deductive databases:

Theorem 2.2 ([MR90]) *For an indefinite deductive database P and a positive clause E , $P \models E$ if and only if $T_P^I \uparrow \omega \models E$ if and only if E is true in every minimal model of P .*

Essentially, $T_P^I \uparrow \omega$ is equivalent to the set of all disjunctions that are true in each minimal model. The bottom-up algebraic method presented in this paper will essentially capture all the disjuncts in $T_P^I \uparrow \omega$.

2.2 C-tables and Disjunctive Facts

The *C-table* structure of [IJ81] is capable of representing more general kinds of incomplete information, but we shall use them to represent disjunctive facts. We shall now define C-tables.

A *domain* is a set of values, usually finite. A *relation scheme* is a list of attribute names, (A_1, \dots, A_n) . We associate a domain with each attribute. Let D_1, \dots, D_n be the domains associated with the attributes A_1, \dots, A_n respectively. Let \mathcal{V} be a set of distinguished variables and \mathcal{C} be a set of distinguished constants. Let us define D_{COND} to be a special domain of logical conditions formed from the elements of \mathcal{V} , \mathcal{C} . We shall associate the domain D_{COND} with a special attribute *COND*. A *C-table* T over the scheme $(R, COND)$, where $R = (A_1, \dots, A_n)$, consists of tuples (t, c) where $t \in D_1 \times \dots \times D_n$ and $c \in D_{COND}$. The tuple (t, c) belonging to the C-table T can be interpreted as the logical formula

$$c \rightarrow T(t)$$

i.e. if the condition c were true then the tuple t belongs to the relation T . The C-table can be used to represent ground disjunctive facts. For example, the ground disjunctive fact $r(t_1) \vee r(t_2)$ can be represented by the two tuples $(t_1, x = a)$ and $(t_2, x \neq a)$ in the C-table T for predicate r . The justification for this is the fact that

$$((x = a) \rightarrow T(t_1)) \wedge ((x \neq a) \rightarrow T(t_2))$$

logically implies $T(t_1) \vee T(t_2)$. In general, a ground disjunctive fact

$$r_1(t_1) \vee \dots \vee r_n(t_n)$$

will be represented by $(n - 1)$ tuples of the form $(t_i, x = a_i)$ in the C-table T_i for predicate r_i , $1 \leq i \leq n - 1$ and a tuple of the form $(t_n, x \neq a_1 \wedge \dots \wedge x \neq a_{n-1})$ in the C-table T_n for predicate r_n . Conversely, if there are n tuples of the form (t_i, c_i) in C-table T_i , $1 \leq i \leq n$, such that $c_1 \vee \dots \vee c_n$ is a tautology, then we say that the ground disjunctive fact $r_1(t_1) \vee \dots \vee r_n(t_n)$ is represented in the C-tables, where T_i is the C-table for the predicate r_i .

Example 2.2 Let us consider the following disjunctive facts:

$$\begin{aligned} &bg(\text{John}, A) \\ &bg(\text{Tom}, A) \vee bg(\text{Tom}, B) \\ &bg(\text{Gary}, A) \vee bg(\text{Gary}, B) \vee bg(\text{Gary}, O) \end{aligned}$$

These disjunctive facts can be represented in the following C-table:

BG		
NAME	BGROUP	COND
John	A	true
Tom	A	$x = a$
Tom	B	$x \neq a$
Gary	A	$y = a$
Gary	B	$y = b$
Gary	O	$(y \neq a) \wedge (y \neq b)$

□

A C-table is said to be *normalized* if

1. it does not contain two tuples (t_1, c_1) and (t_2, c_2) with $t_1 = t_2$ and
2. it does not contain a tuple of the form (t, c) , where c is a contradiction.

To normalize a C-table, we simply delete all tuples of the form (t, c) , where c is a contradiction. and combine the tuples $(t, c_1), \dots, (t, c_k)$ into one tuple $(t, c_1 \vee \dots \vee c_k)$ (It can be easily observed that the logical formula $(c_1 \rightarrow T(t)) \wedge \dots \wedge (c_k \rightarrow T(t))$ is logically equivalent to $(c_1 \vee \dots \vee c_k \rightarrow T(t))$). We shall assume that all C-tables are normalized. Often, we shall replace tuple (t, c) by (t, c') where c and c' are equivalent. If T were a C-table then we denote its normalized form by T^* .

We now define the relevant extended relational algebraic operations for C-tables.

Selection Let T be a C-table defined on the scheme $(R, COND)$ and let F be a selection formula involving the attributes of R . Then,

$$\sigma_F(T_1) = \{t | (\exists t_1)(t_1 \in T_1 \wedge t[R] = t_1[R] \wedge t[COND] = (t_1[COND] \wedge F(t_1)))\}^*$$

where $F(t_1)$ is F with all occurrences of attribute A replaced by $t_1[A]$. An example of the selection operation is shown below:

T		
A	B	$COND$
a_1	b_1	$true$
a_2	b_1	$x = a$
a_2	b_2	$x \neq a$
a_3	b_2	$y = a$
a_3	b_3	$y \neq a$

$\sigma_{(B=b_1) \vee (B=b_2)}(T)$		
A	B	$COND$
a_1	b_1	$true$
a_2	b_1	$x = a$
a_2	b_2	$x \neq a$
a_3	b_2	$y = a$

Projection Let T be a C-table defined on the scheme $(R, COND)$ and let Y be a list of attributes of R . Then,

$$\Pi_Y(T_1) = \{t | (\exists t_1)(t_1 \in T_1 \wedge t[Y] = t_1[Y] \wedge t[COND] = t_1[COND])\}^*$$

An example of the projection operation is shown below:

T		
A	B	$COND$
a_1	b_1	$true$
a_2	b_1	$x = a$
a_2	b_2	$x \neq a$
a_2	b_3	$y = a$
a_3	b_3	$y \neq a$
a_4	b_1	$z = a$
a_5	b_1	$z \neq a$

$\Pi_A(T)$	
A	$COND$
a_1	$true$
a_2	$true$
a_3	$y \neq a$
a_4	$z = a$
a_5	$z \neq a$

Join Let T and W be two C-tables defined on the schemes $(R, COND)$ and $(S, COND)$ respectively. Then,

$$T \bowtie W = \{t \mid (\exists t_1)(\exists t_2)(t_1 \in T \wedge t_2 \in W \wedge t[R] = t_1[R] \wedge t[S - R] = t_2[S - R] \wedge t[COND] = (t_1[COND] \wedge t_2[COND] \wedge \bigwedge_{A \in R \cap S} (t_1[A] = t_2[A])))\}^*$$

An example of the join operation is shown below:

T		
A	B	$COND$
a_1	b_1	$true$
a_2	b_2	$x = a$
a_3	b_2	$x \neq a$
a_4	b_3	$y = a$
a_4	b_4	$y \neq a$

W		
B	C	$COND$
b_1	c_1	$true$
b_1	c_2	$z = a$
b_1	c_3	$z \neq a$
b_2	c_4	$w = a$
b_3	c_5	$w \neq a$

$T \bowtie W$			
A	B	C	$COND$
a_1	b_1	c_1	$true$
a_1	b_1	c_2	$z = a$
a_1	b_1	c_3	$z \neq a$
a_2	b_2	c_4	$x = a \wedge w = a$
a_3	b_2	c_4	$x \neq a \wedge w = a$
a_4	b_3	c_5	$y = a \wedge w \neq a$

Union Let T_1 and T_2 be two C-tables defined on the scheme $(R, COND)$. Then,

$$T_1 \cup T_2 = \{t | t \in T_1 \vee t \in T_2\}^*$$

An example of the union operation is given below:

T_1	
A	$COND$
a_1	$true$
a_2	$x = a$
a_3	$x \neq a$
a_4	$y = a$
a_5	$y = b$
a_6	$y \neq a \wedge y \neq b$

T_2	
A	$COND$
a_2	$true$
a_4	$z = a$
a_5	$z \neq a$
a_6	$w = a$

$T_1 \cup T_2$	
A	$COND$
a_1	$true$
a_2	$true$
a_3	$x \neq a$
a_4	$(y = a) \vee (z = a)$
a_5	$(y = b) \vee (z \neq a)$
a_6	$(y \neq a \wedge y \neq b) \vee (w = a)$

3 BOTTOM-UP APPROACH

In this section, we present a bottom-up algorithm to evaluate non-Horn clauses in an indefinite deductive database. The projection operation is further generalized to be able to produce disjunctive facts that can be derived from the multiple atoms in the head of non-Horn clauses. Such a generalization is called *project-or*. We then define IDB equations which are based on the algebraic operations on C-tables. An algorithm to solve the IDB equations is presented. The solution to the IDB equations of an indefinite deductive database correspond to their fixpoint semantics.

3.1 Project-Or operator

Consider the following non-Horn clause:

$$p(x, y), q(x, z) \leftarrow r(x, z), s(z, y)$$

The extended relational algebraic operations defined earlier can be used to compute the C-table that corresponds to the body of the clause. The following algebraic expression corresponds to the C-table, $BODY$ for the body of the clause:

$$BODY(X, Y, Z) = \Pi_{X, Y, Z}(R(X, Z) \bowtie S(Z, Y)),$$

where R and S are the M-tables for the predicates r and s respectively. If this were a Horn clause with only one atom in the head, we could use an appropriate projection operation to compute the C-table for the head predicate. However, since we have two atoms in the head of the clause, we need to further extend the projection operation. Such an operation, which we shall term *project-or*, must take in as input a C-table (one that corresponds to the body of the clause) and projection attribute lists, one for each atom in the head and return as output C-tables, one for each distinct head predicate. We shall use the symbol Π for project-or. In the example clause, the project-or operation is

$$P, Q = \Pi_{\{(X,Y)\},\{(X,Z)\}}(BODY(X, Y, Z)).$$

This operation will return two C-tables P and Q . Let us now discuss what the contents of P and Q should be. Suppose that the C-table for the body of the clause is

<i>BODY</i>			
<i>X</i>	<i>Y</i>	<i>Z</i>	<i>COND</i>
a_1	b_1	c_1	<i>true</i>
a_2	b_2	c_2	$x = a$
a_2	b_3	c_3	$x \neq a$

The project-or should produce the following C-tables:

<i>P</i>			<i>Q</i>		
<i>X</i>	<i>Y</i>	<i>COND</i>	<i>X</i>	<i>Z</i>	<i>COND</i>
a_1	b_1	$y = a$	a_1	c_1	$y \neq a$
a_2	b_2	$x = a \wedge u = a$	a_2	c_2	$x = a \wedge u \neq a$
a_2	b_3	$x \neq a \wedge v = a$	a_2	c_3	$x \neq a \wedge v \neq a$

where y , u , and v are newly generated variables used to express the disjunctions which can be derived by using the non-Horn clause and the tuples of *BODY*.

We shall now present a formal operational definition of the project-or operation.

Project-or Let T be an input C-table defined on the scheme $(R, COND)$ and let Y_1, \dots, Y_n be a list of sets of projection attribute lists made up of attributes of R . Some of the Y_i s may be empty sets. Let X_1, \dots, X_m be the non-empty sets among Y_1, \dots, Y_n such that $X_i = Y_{k_i}$, $1 \leq i \leq m$. Let the order of the Y_i s be maintained among X_i s, i.e. $i \leq j$ iff $k_i \leq k_j$, $1 \leq i, j \leq m$. If $m = 1$ and if X_m is a singleton, then the project-or operation simply reduces to the projection operation. So, we shall assume for the remaining of this definition that $m > 1$. The sets Y_1, \dots, Y_n correspond to the output C-tables T_1, \dots, T_n respectively. The T_i s are computed as follows:

Case 1: $Y_i = \emptyset$ In this case $T_i = \emptyset$.

Case 2: $Y_i \neq \emptyset$ Let $X_i = \{X_{i_1}, \dots, X_{i_{p_i}}\}$, $1 \leq i \leq m$. For each tuple (t, c) in T ,

1. introduce tuples $(\Pi_{X_{i_1}}(t), c \wedge x = a_{i_1}), \dots, (\Pi_{X_{i_{p_i}}}(t), c \wedge x = a_{i_{p_i}})$ in C-table T_{k_i} , $1 \leq i < m$,
2. introduce tuples $(\Pi_{X_{m_1}}(t), c \wedge x = a_{m_1}), \dots, (\Pi_{X_{m(p_{m-1})}}(t), c \wedge x = a_{m(p_{m-1})})$ in C-table T_{k_m} , and

3. introduce the tuple $(\Pi_{X_m p_m}(t), c \wedge x \neq a_{11} \wedge \dots \wedge x \neq a_{m(p_m-1)})$ in C-table T_{k_m} ,

where x is a unique variable symbol of \mathcal{V} and $a_{i,j}$ s are different constants from \mathcal{C} . Finally, the resulting C-tables are normalized. An example of the project-or operation is shown below:

$$P, Q = \Pi_{\{(1,2),(1,3)\},\{(2,3)\}}(T)$$

a_1	b_1	c_1	$true$
a_2	b_2	c_2	$x = a$
a_2	b_3	c_3	$x \neq a$

a_1	b_1	$y = a$
a_1	c_1	$y \neq a \wedge y \neq b$
a_2	b_2	$x = a \wedge u = a$
a_2	c_2	$x = a \wedge u \neq a \wedge u \neq b$
a_2	b_3	$x \neq a \wedge v = a$
a_2	c_3	$x \neq a \wedge v \neq a \wedge v \neq b$

b_1	c_1	$y = b$
b_2	c_2	$x = a \wedge u = b$
b_3	c_3	$x \neq a \wedge v = b$

3.2 IDB Equations and their Solution

Let $DB = EDB \cup IDB$ be an indefinite deductive database where the EDB part is represented as C-tables and the IDB part consists of non-Horn rules. We can partition the set of IDB predicates based on the equivalence relation “predicate p is related to predicate q if and only if p and q both appear in the head of the same non-Horn rule”. We can use this partition to define a partition of the set of non-Horn rules in IDB based on the equivalence relation “non-Horn rule r_1 is related to non-Horn rule r_2 if and only if there exists head predicate p_1 in r_1 and head predicate p_2 in r_2 such that p_1 and p_2 appear in the same equivalence class of predicates”.

Let $\Pi = \{\Pi_1, \dots, \Pi_n\}$ be the partition of the set of non-Horn rules of the IDB . We shall obtain an IDB equation for each element of this partition. Let $\Pi_i = \{r_1, \dots, r_k\}$ define the predicates p_{i_1}, \dots, p_{i_k} (the head predicates in the rules of Π_i) and let $r \in \Pi_i$ be the following non-Horn rule:

$$p_1(X_1), \dots, p_m(X_m) \leftarrow q_1(Y_1), \dots, q_n(Y_n)$$

The algebraic expression for the body of r can be obtained in a straightforward way using the algorithm presented in Chapter 3 of [Ull88]. Let $EVALRULE(r, Q_1, \dots, Q_n)$ be the algebraic expression for the body of r , where Q_i is the C-table for the predicate q_i . Then, the algebraic expression for the rule r is

$$E(r) = \Pi_{Z_1, \dots, Z_i}(EVALRULE(r, Q_1, \dots, Q_n))$$

where $Z_i = \emptyset$ if p_i does not appear in the head of the rule and $Z_i = \{X_{j_1}, \dots, X_{j_s}\}$ if p_i appears in the head of r as p_{j_1}, \dots, p_{j_s} . The IDB equation for Π_i is

$$P_{i_1}, \dots, P_{i_k} = \bigcup_{i=1}^k E(r_i).$$

Remark: The union in this equation is a natural generalization of the union of C-tables and is defined as follows:

$$R_1, \dots, R_k \cup S_1, \dots, S_k = R_1 \cup S_1, \dots, R_k \cup S_k$$

where R_i and S_i are union compatible C-tables, $1 \leq i \leq k$.

We shall write $\bigcup_{i=1}^k E(r_i)$ as $EVAl(\Pi_i, P_1, \dots, P_m, R_1, \dots, R_n)$, where P_1, \dots, P_m are the C-tables for the IDB predicates and R_1, \dots, R_n are the EDB C-tables.

Example 3.1 Let the IDB part contain the following rules:

$$\begin{aligned} r_1: & p_1(x, y), p_2(x, z) \leftarrow q(x, y), r(y, z) \\ r_2: & p_2(x, y) \leftarrow q(x, z), p_4(y) \\ r_3: & p_1(x, y), p_3(z) \leftarrow r(x, y), s(y, z) \\ r_4: & p_4(x) \leftarrow q(x, y) \\ r_5: & p_4(x), p_5(y) \leftarrow r(x, y), p_4(y) \\ r_6: & p_6(x, y), p_7(x), p_6(y, x) \leftarrow r(x, z), p_2(z, y) \\ r_7: & p_7(x) \leftarrow p_6(x, y) \end{aligned}$$

The partition of the set of IDB predicates is $\{\{p_1, p_2, p_3\}, \{p_4, p_5\}, \{p_6, p_7\}\}$ and the partition of the set of rules is $\{\{r_1, r_2, r_3\}, \{r_4, r_5\}, \{r_6, r_7\}\}$. The IDB equations are

$$\begin{aligned} P_1, P_2, P_3 &= \Pi_{\{(X,Y)\}, \{(X,Z)\}, \emptyset}(Q(X, Y) \bowtie R(Y, Z)) \cup \\ &\quad \Pi_{\emptyset, \{(X,Y)\}, \emptyset}(Q(X, Z) \bowtie P_4(Y)) \cup \\ &\quad \Pi_{\{(X,Y)\}, \emptyset, \{(Z)\}}(R(X, Y) \bowtie S(Y, Z)) \\ P_4, P_5 &= \Pi_{\{(X)\}, \emptyset}(Q(X, Y)) \cup \\ &\quad \Pi_{\{(X)\}, \{(Y)\}}(R(X, Y) \bowtie P_4(Y)) \\ P_6, P_7 &= \Pi_{\{(X,Y), (Y,X)\}, \{(X)\}}(R(X, Z) \bowtie P_2(Z, Y)) \cup \\ &\quad \Pi_{\emptyset, \{(X)\}}(P_6(X, Y)) \end{aligned}$$

□

We shall now present an algorithm to solve a set of IDB equations. Let p_1, \dots, p_m be the IDB predicates of an indefinite deductive database and P_1, \dots, P_m be the corresponding C-tables. Also let R_1, \dots, R_n be the EDB C-tables of the database. Suppose that the partition of the non-Horn rules in IDB is $\{\Pi_1, \dots, \Pi_k\}$. We shall denote the predicates defined in partition Π_i by $p_{i_1}, \dots, p_{i_{n_i}}$, $1 \leq i \leq k$. The algorithm to compute a solution to the IDB equations of an indefinite deductive database is given in Figure 1.

Example 3.2 Let us reconsider the disjunctive database of Example 2.1. The extensional database can be represented in the following C-table:

a_4	a_6	$true$
a_1	a_3	$x = a$
a_3	a_4	$x \neq a$

```

for  $i \leftarrow 1$  to  $m$  do  $P_i = \emptyset$ 
repeat
  for  $i \leftarrow 1$  to  $m$  do  $Q_i = P_i$ 
  for  $i \leftarrow 1$  to  $k$  do
     $P_{i_1}, \dots, P_{i_n} \leftarrow EVAL(\Pi_i, Q_1, \dots, Q_m, R_1, \dots, R_n)$ 
until  $Q_i = P_i$ , for all  $i$ ,  $1 \leq i \leq m$ 
output  $Q_1, \dots, Q_m$ 

```

Figure 1: Algorithm to solve IDB equations

The IDB equations for the non-Horn rules are:

$$\begin{aligned}
R4 &= \Pi_{\{(X,Y)\}}(R3(X,Y)) \cup \Pi_{\{(X,Y)\}}(R3(X,Z) \bowtie R4(Z,Y)) \\
R1, R2 &= \Pi_{\{(X)\}, \emptyset}(R3(Y,X)) \cup \Pi_{\{(X)\}, \{(Y)\}}(R4(X,Y))
\end{aligned}$$

Computing the solution to these equations, we obtain the following normalized C-tables:

a_4	a_6	$true$
a_1	a_3	$x = a$
a_3	a_4	$x \neq a$
a_3	a_6	$x \neq a$

a_6	$true$
a_3	$(x = a) \vee (u = a) \vee (v = a)$
a_4	$(x \neq a) \vee (y = a)$
a_1	$x = a \wedge z = a$

a_6	$(y \neq a) \vee (x \neq a \wedge v \neq a)$
a_3	$(x = a \wedge z \neq a)$
a_4	$(x \neq a \wedge u \neq a)$

Let us confirm if this solution indeed corresponds to $T_P^I \uparrow \omega$ of Example 2.1. Consider the tuples $(a_1, x = a \wedge z = a) \in R1$, $(a_3, x = a \wedge z \neq a) \in R2$, and $((a_3, a_4), x \neq a) \in R4$. Since $(x = a \wedge z = a) \vee (x = a \wedge z \neq a) \vee (x \neq a)$ is a tautology, we conclude that $r_1(a_1) \vee r_2(a_3) \vee r_4(a_3, a_4)$ is represented in the C-tables. A tedious verification will indicate that for every set of tuples in the C-tables such that the disjunction of their conditions is a tautology, we can find a disjunct in $T_P^I \uparrow \omega$ that is represented by this set of tuples. Conversely, consider the disjunct $r_1(a_3) \vee r_2(a_4)$ in $T_P^I \uparrow \omega$. The tuples that correspond to this disjunct are $(a_3, (x = a) \vee (u = a) \vee (v = a)) \in R1$ and $(a_4, x \neq a \wedge u \neq a) \in R2$. The disjunction of the conditions $((x = a) \vee (u = a) \vee (v = a)) \vee (x \neq a \wedge u \neq a)$ is a tautology and hence the disjunct $r_1(a_3) \vee r_2(a_4)$ is represented in the C-tables. Once again, a tedious verification will indicate that every disjunct in $T_P^I \uparrow \omega$ is represented in the C-tables. \square

4 CURRENT STATE OF KNOWLEDGE

As has been noted earlier, most of the research on implementation issues of deductive databases has concentrated on definite deductive databases. A comprehensive discussion of these results

can be found in [Ull88]. As far as indefinite deductive databases and their implementations are concerned, one can find relatively few scattered work. We shall mention some of the bottom-up methods for indefinite deductive databases.

Henschen and Park [HP88] provide results with respect to yes/no answers to queries posed over indefinite deductive databases. They handle negation by using the Generalized Closed World Assumption (GCWA). They present several fundamental results on compiling the GCWA in indefinite deductive databases. They also present three representation schemes which separate the rules from the facts. Using these schemes, they isolate the deduction part in answering queries from the retrieval part. Several effective ways for compiling the GCWA inference on the rules and evaluating it through the facts are presented for non-recursive databases. Recursive rules are not adequately treated.

Grant and Minker [GM86] have developed algorithms to answer arbitrary queries in indefinite databases. They provide algorithms to check if a candidate answer is indeed an answer to the query. Using this algorithm, they present an algorithm to find all minimal answers to queries. Although this paper does not deal with rules, queries can be answered by straightforward extensions to the algorithms.

Imieliński's C-tables [IJ81] are capable of representing disjunctive facts. The extended relational algebra can be used to answer queries in indefinite databases without rules.

Liu and Sunderraman [LS91] present a generalization to the relational model to represent disjunctive facts in tabular structures called M-tables. Queries can be answered using the generalized relational algebra. In [LS90], they apply the generalized model to indefinite deductive databases.

References

- [GM86] John Grant and Jack Minker. Answering queries in indefinite databases and the null value problem. In *Advances in Computing Research, Volume 3*, pages 247-267. JAI Press Inc., 1986.
- [GM89] John Grant and Jack Minker. Deductive database theories. UMIACS-TR-89-77, CS-TR-2293, Department of Computer Science, University of Maryland, College Park, MD 20742, 1989.
- [HP88] Lawrence Henschen and Hyoung-Sik Park. Compiling the GCWA in indefinite deductive databases. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 395-438. Morgan Kaufmann, Los Altos, CA, 1988.
- [IJ81] Tomasz Imieliński and Witold Lipski Jr. On representing incomplete information in a relational database. In *Proceedings of the 7th International Conference on Very Large Data Bases, Cannes, France*, pages 389-397, New York, September 1981. IEEE.
- [LS90] Ken-Chih Liu and Rajshekhar Sunderraman. An algebraic approach to indefinite deductive databases. In *Proceedings of the 5th International Symposium on Methodologies for Intelligent Systems, Knoxville, Tennessee*, New York, October 1990. Elsevier Press.

- [LS91] Ken-Chih Liu and Rajshekhar Sunderraman. A generalized relational model for indefinite and maybe information. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):65–77, 1991.
- [Min82] Jack Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science, N138*, pages 292–308. Springer-Verlag, 1982.
- [MR90] Jack Minker and Arcot Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9:45–74, 1990.
- [MUG86] K. Morris, J.D. Ullman, and A. Van Gelder. Design overview of the NAIL! system. In E. Shapiro, editor, *Proceedings of the Third International Conference on Logic Programming*, pages 554–568. Springer-Verlag, New York, 1986.
- [NT88] Shamim Naqvi and Shalom Tsur. *A Logic Language for Data and Knowledge Bases*. Computer Science Press, Rockville, MD, 1988.
- [SR86] Michael Stonebraker and Lawrence Rowe. The POSTGRES papers. UCB/ERL M86/85, Dept. of EECS, University of California at Berkeley, 1986.
- [Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I and II. Computer Science Press, Rockville, Maryland, 1988.

INHERITANCE IN CONCEPTUAL NETWORKS*

Leixuan Yang

Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5

Stan Szpakowicz**

Computer Science Department
University of the Witwatersrand
Johannesburg 2050, South Africa

ABSTRACT

We present the inheritance system in a knowledge representation formalism called Conceptual Network. Exceptions can be represented in this formalism by means of conditions attached to conceptual relations. The explicit representation of exceptions allows us to implement monotonic inference for the inheritance problem. We look at the way in which our inheritance system works on some of the classical examples. This system offers a new approach to the inheritance problem, with a new representation method and a simple monotonic inference algorithm.

1. INTRODUCTION

An inheritance system is a representation system founded on the hierarchical structuring of knowledge¹⁰. This structuring is known in Artificial Intelligence as the inheritance hierarchy. Methods of representing inheritance hierarchies include Fahlman's NETL system^{4,5}, Etherington and Reiter's system—the default logic approach^{2,3}, Padgham's lattice-based model⁶, and Shastri's evidential formalization^{7,8}. These systems all have their limitations. For instance, in NETL the inferences are performed in parallel. As Etherington and Reiter note, "there is the unfeasibility of completely general massively parallel architectures for dealing with inheritance with exceptions"³. They formulate the problem of inheritance in terms of default logic, and provide a non-monotonic inference method.

In this paper, we present the inheritance system in a knowledge representation system called Conceptual Network (in short: CN). The representation of inheritance hierarchies in CN allows us to implement a monotonic inference method for dealing with the problem of inheritance. This offers a new approach to the inheritance problem. We first briefly introduce CN. Next, the representation of inheritance hierarchies in CN is presented. We give a monotonic inference algorithm for inheritance with exceptions, and apply it to some classical examples. All of the ideas presented in this paper are now being implemented.

* This work was supported by the Natural Sciences and Engineering Research Council of Canada, and by Cognos Inc.

** On leave from the University of Ottawa

2. CONCEPTUAL NETWORK

The Conceptual Network formalism^{12, 13, 14} was developed with TANKA, a system for semi-automatic knowledge acquisition by text analysis⁹, and it is used there to represent knowledge. TANKA will process technical text, and incrementally build a conceptual network which models the domain; text understanding will be turned into knowledge acquisition.

There are four basic elements in CN: concepts, relations, structures and conditions. Knowledge is represented by a combination of these four elements. A concept represents a number of instances in the world. We use $I(c)$ to denote the set of instances of concept c . Concept $c1$ is called a subclass of concept $c2$ (and $c2$ a superclass of $c1$) if and only if $I(c1)$ is a subset of $I(c2)$. Concepts are broadly classified into three groups: objects, activities, and properties. Relations capture the relationships that hold between concepts. CN only supports binary relations. Two hierarchical relations, specialization *is_a* and generalization *kind*, are used to build the inheritance hierarchies.

Conditions represent contextual restrictions on the concepts involved in a relation. Let R be a relation between concepts $c1$ and $c2$. A condition attached to $R(c1, c2)$ can be regarded as the description of a subclass $c1'$ of $c1$ and a subclass $c2'$ of $c2$: only $c1'$ and $c2'$ can be in the relation R . The condition is a logical expression built of simple conditions. We will not give the precise definition of simple conditions in this paper, because we have only special forms of simple conditions in the hierarchies. The following are two examples of simple conditions:

```
elephant isnot royal_elephant
```

describes the set of instances of *elephant*, $\{i \mid i \in I(\textit{elephant}) \setminus I(\textit{royal_elephant})\}$ and

```
elephant is royal_elephant
```

describes the set of instances of *elephant*, $\{i \mid i \in I(\textit{elephant}) \cap I(\textit{royal_elephant})\}$. We say that a simple condition *Cond* is satisfied by an instance i if and only if i is in the set of instances described by *Cond*; *Cond* is unsatisfied by i if and only if i is not in the set of instances described by *Cond*.

Structures are used to represent complex concepts, composed of other concepts. The available structures are *individual*, *sequence*, *collection*, *tuple*, *union*, and *intersection*.

In the textual notation for CN, every concept is associated with a cluster of concepts directly linked to it by binary relations. We call such a cluster a *frame* of this concept. A frame is notated as a group of slots, each describing one link. The format of a frame of concept $c0$ is:

```
type_of_c0 c0.
slots.
end c0.
```

where *type_of_c0* is OBJECT, ACTIVITY OR PROPERTY. There must be at least one slot in a frame. Each slot denotes a relation between $c0$ and another concept. If the relation is a hierarchical relation (*is_a*, *kind*), the slot is called a hierarchical slot. The format of a hierarchical slot is:

```
[condition] relation_name: (structure_of_ci) ci
```

where `relation_name` must be `is_a` or `kind`. Such slots describe a hierarchical relationship between concept `c0`, whose type is `type_of_c0`, and concept (structure_of_ci) `ci` of the same type. If `structure_of_ci` is not `individual`, (structure_of_ci) `ci` represents a complex concept. The default is `individual`: (individual) `ci` is the same as `ci`. The default for condition is `true`. We introduce conditions into hierarchical slots to deal with exceptions—this will be shown later.

3. REPRESENTING INHERITANCE HIERARCHIES IN CN

To deal with inheritance with exceptions, Etherington and Reiter have identified 5 link types³, used to represent the inheritance hierarchies. In CN these links are represented as follows:

1) Strict IS-A

```
type_of_c1 c1.
is_a: c2.
.....
end c1.
```

c1 is *c2*, that is, any instance of *c1* is always an instance of *c2*.

2) Strict IS-NOT-A

```
type_of_c1 c1.
[false] is_a: c2.
.....
end c1.
```

c1 is not *c2*, that is, no instance of *c1* is an instance of *c2*. No instance can satisfy the false condition. For a given instance *i*, if a condition does not hold for *i*, it is called an *unsatisfied* condition. Unsatisfied conditions are used to block inheritance from the superclass—this will be discussed in detail in the next section.

3) Default IS-A

```
type_of_c1 c1.
[this_frame isnot ce] is_a: c2.
.....
end c1.
```

Normally *c1* is *c2*, but there may be exceptions. In other words, any instance in $I(c1) \setminus I(ce)$ is an instance of *c2*. We should note that only the instances in $I(c1) \cap I(ce)$, rather than all the instances $I(ce)$, are not the instances of *c2*. When *ce* is a subclass of *c1*, the condition implies that any *ce* is not *c2*.

4) Default ISN'T-A

```
type_of_c1 c1.
[this_frame is ce] is_a: c2.
.....
end c1.
```

Normally $c1$ is not $c2$, but there may be exceptions. In other words, instances of $c1$, excluding instances of ce , are not instances of $c2$. Thus, the instances in $I(c1) \cap I(ce)$ are $c2$. When ce is a subclass of $c1$, this condition implies that ce is $c2$.

5) Exception

The exception link in Etherington and Reiter's method is already represented in the conditions of our representation of default IS-A and ISN'T-A. Actually, exception links in their method are always attached to other links, and they describe exceptions to the knowledge represented by those links. That is why we do not have to represent them separately.

Without the loss of generality, we consider only single exceptions. If there were more, we would use complex conditions. A condition of the form

```
[this_frame isnot ce1 AND this_frame isnot ce2 AND ...]
```

with only the **AND** operator is for default IS-A ($ce1, ce2, \dots$ are exceptions). A condition of the form

```
[this_frame is ce1 OR this_frame is ce2 OR ...]
```

with only the **OR** operator is for default ISN'T-A.

We illustrate our representation with the following example ¹¹:

F1: Elephants are grey things.

F2: Royal elephants are elephants.

F3: Royal elephants are not grey things.

Our representation of these facts is:

```
OBJECT elephant.
[this_frame isnot royal_elephant] is_a: grey_thing
end elephant.

OBJECT royal_elephant.
is_a: elephant.
end royal_elephant.
```

In Etherington and Reiter's approach, assertions F1 - F3 would be represented as the following default rule D1, and first-order assertions A1, A2:

```
elephant(x) : NOT royal-elephant(x)
D1: -----
      grey-thing(x)

A1: (x) royal-elephant(x) --> NOT grey-thing(x)
A2: (x) royal-elephant(x) --> elephant(x)
```

Notice that we must encode F3 in the assertion A1 in spite of the fact that this information is already implicit in the information encoded in A2 and D1. If *royal-elephant* is a subclass of *elephant* and elephants other than royal elephants are grey, then it follows that royal elephants are not grey. The CN representation overcomes this drawback—no explicit

relation is required to encode F3, which is captured in the condition of the relation $is_a(elephant, grey_thing)$ and the hierarchical relation $is_a(royal_elephant, elephant)$. The condition in the first frame shows that *elephant* is *grey_thing* unless it is *royal_elephant*, and the hierarchical relation in the second frame means that each *royal_elephant* is *elephant*. The fact that *royal_elephant* is not *grey_thing* is captured by both this condition and this hierarchical relation (see the CN representation of 'default IS-A').

4. MONOTONIC INFERENCE

For a given instance i , a concept c is called a positive concept of i if and only if it has i as its instance. c is called a negative concept of i if and only if it does not have i as its instance. We are concerned with the following inheritance problem:

Given are two sets of concepts, $P = \{cp_1, cp_2, \dots, cp_k\}$, and $N = \{cn_1, cn_2, \dots, cn_m\}$. Concepts in P are positive concepts of an instance i . Concepts in N are negative concepts of i . We have two inference rules.

Rule1

$c1$ is a positive concept of i & $is_a(c1, c2)$ &
 i satisfies the condition of $is_a(c1, c2)$
 $\Rightarrow c2$ is a positive concept of i

Rule2

$c1$ is a positive concept of i & $is_a(c1, c2)$ &
 i does not satisfy the condition of $is_a(c1, c2)$
 $\Rightarrow c2$ is a negative concept of i

Let H be the set of all given hierarchical relations. We need to derive the theorems of the theory $\langle H \cup P \cup N, \{Rule1, Rule2\} \rangle$, where the first element represents axioms and the second one represents inference rules. All these derived theorems together are called the extension of i . From the inference rules, we can see that a subclass $c1$ can be extended to its superclass $c2$ only if $c1$ is a positive concept of the given instance. No extension can be made from the negative concepts, which are only considered in the termination of extensions.

Etherington and Reiter's method allows extensions to be constructed by a series of successive approximations. The previous approximations may be overridden by the current one. When two successive approximations are the same, the procedure is said to converge and the extension is the current approximation. This reasoning procedure is non-monotonic, in the sense that new information can invalidate previously derived facts. In this section, we present a monotonic inference method for finding extensions.

The main idea that underlies the monotonic inference method is to derive new facts only if the truth values of the conditions of these facts have been determined. If the truth value of a condition cannot be determined when we check it, this condition is *undetermined* and will be suspended until new derived facts make it determined. This means that we always reason with certainty, so that new facts will not invalidate the previously derived facts. In this sense, we say that our inference method is monotonic.

We first define a few simple auxiliary operations. S denotes a set, $C1, C2$ are concepts:

```
remove(S):      remove from S and return a random element e
add(S, e):      add an element e to S
empty(S):       return true if S is empty, otherwise false
member(e, S):   return true if e is an element of S, otherwise false
condition(C1, C2): returns the condition of is_a(C1, C2)
```

The inference algorithm is described in pseudocode. The set of concepts not yet considered for extension is maintained in the variable `CSet` (this stands for “concept-set”). Initially, it contains the positive concepts. Two concept sets, `PE` and `NE`, initialized with the original sets of positive and negative concepts, are gradually extended by the algorithm. There is a main procedure, and two subroutines. Subroutine `check-cond(Cond, i, Result)` tests whether condition `Cond` is satisfied by instance i . If yes, the extension can proceed from the related concept by adding to the `CSet`. If not, no extension can be made from the related concept. If the condition `Cond` is undetermined, it will be suspended. The subroutine applies negation-by-failure: if it cannot be derived that i is an instance of a concept c , it is assumed that i is not an instance of c .

We only consider simple conditions in this algorithm. Complex conditions are composed of simple conditions by means of the operators **AND**, **OR**. The fact that a complex condition `CCond` is satisfied by an instance i can be determined by applying `check-cond(, ,)` to the simple conditions of `CCond`. If `CCond` is composed of simple conditions by the operator **AND**, then apply `check-cond(, ,)` to a simple condition of `CCond` each time; repeat this process until a simple condition of `CCond` is unsatisfied or no more simple conditions remain to be checked. If `CCond` is composed of simple conditions by the operator **OR**, then apply `check-cond(, ,)` to a simple condition of `CCond` each time; repeat until a simple condition of `CCond` is satisfied or no more simple conditions remain. Whenever a new fact is derived, the suspended conditions will be examined to see if any condition can be determined now. This is done by subroutine `re-eval-cond(Sign, Concept)`, where `Sign` marks a concept as positive or negative. When the `CSet` is empty, that is, no more extensions can be made, the main procedure is terminated. `NE` and `PE` together form the extension of i .

The Inference Algorithm

Given: an instance i , a set $P = \{ cp_1, cp_2, \dots, cp_k \}$ of positive concepts of i , and a set $N = \{ cn_1, cn_2, \dots, cn_m \}$ of negative concepts of i .

Sought: `PE` and `NE`—positive and negative extensions.

```
PE := P;          -- initialize PE
NE := N;          -- initialize NE
Susp := {};       -- initialize Susp (will store suspended conditions)
CSet := P;        -- initialize CSet (will store concepts
                  -- from which further extensions may be made)
repeat           -- for every element in CSet:
  C := remove(CSet);
                -- get one concept and try to make extension from it
  ParentL := the list of all parents of C in the is_a hierarchy;
```

```

for Cnp in ParentL do      -- for every parent of C:
  Cond := condition(C, Cnp);
  if not member(Cnp, PE) & not member(Cnp, NE) then
    check-cond(Cond, i, Result);
    if Result = satisfied then
      add(PE, Cnp);          -- Cnp is a positive concept of i
      add(CSet, Cnp);        -- further extension may be made from Cnp
      re-eval-cond(pos, Cnp); -- try to "unsuspend"
    elsif Result = unsatisfied then
      add(NE, Cnp);          -- Cnp is a negative concept of i
      re-eval-cond(neg, Cnp); -- try to "unsuspend"
    elsif Result = undetermined & Cond = [this_frame OP Ce] then
      -- (OP is the relation operator in the condition)
      add(Susp, undet-cond(Cnp, Ce, Cond));
      -- suspend the undetermined condition
    endif;
  endif;
endfor;
until empty(CSet);

procedure check-cond(Cond, i, Result);
  -- is condition Cond satisfied for instance i?
  -- Result: satisfied, unsatisfied, undetermined
if Cond = [true] then Result := satisfied;
elsif Cond = [false] then Result := unsatisfied;
elsif Cond = [this_frame isnot Ce] then
  if member(Ce, NE) then Result := satisfied;
  elsif member(Ce, PE) then Result := unsatisfied;
  else
    if no subclass of Ce is in the CSet then
      Result := satisfied; -- we cannot determine whether i is an
                          -- instance of Ce: negation by failure
    else Result := undetermined;
    endif;
  endif;
elsif Cond = [this_frame is Ce] then
  if member(Ce, NE) then Result := unsatisfied;
  elsif member(Ce, PE) then Result := satisfied;
  else
    if no subclass of Ce is in the CSet then
      Result := unsatisfied;
    else Result := undetermined;
    endif;
  endif;
endif;

```

```

procedure re-eval-cond(Sign, Cnp);
    -- re-evaluate suspended conditions
remove all undet-cond(Cnp, Cel, Condl), ...,
    undet-cond(Cnp, Cej, Condj) from Susp;
-- it was known before this subroutine has been called whether i is
-- an instance of Cnp; therefore all suspended conditions of the form
-- undet-cond(Cnp, _, _) will no longer need to be re-evaluated
Re-eval-set :=
    all suspended conditions of the form undet-cond(_, Cnp, _);
Susp := Susp \ Re-eval-set;
case Sign in
pos:          -- Cnp is a positive concept of i
  for RC in Re-eval-set do
    if RC = undet-cond(Cg, Cnp, [this_frame isnot Cnp]) &
      not member(Cg, NE) then
      add(NE, Cg);          -- Cg is a negative concept of i
      re-eval-cond(n, Cg);  -- try to "unsuspend" more
    elsif RC = undet-cond(Cg, Cnp, [this_frame is Cnp]) &
      not member(Cg, PE) then
      add(PE, Cg);          -- Cg is a positive concept of i
      add(CSet, Cg);
      re-eval-cond(p, Cg);  -- try to "unsuspend" more
    endif;
  endfor;
neg:          -- Cnp is a negative concept of i
  for RC in Re-eval-set do
    if RC = undet-cond(Cg, Cnp, [this_frame isnot Cnp]) &
      not member(Cg, PE) then
      add(PE, Cg);          -- Cg is a positive concept of i
      add(CSet, Cg);
      re-eval-cond(p, Cg);  -- try to "unsuspend" more
    elsif RC = undet-cond(Cg, Cnp, [this_frame is Cnp]) &
      not member(Cg, NE) then
      add(NE, Cg);          -- Cg is a negative concept of i
      re-eval-cond(n, Cg);  -- try to "unsuspend" more
    endif;
  endfor;
endcase;

```

In this algorithm, we can see that only positive concepts can occur in the concept-set (CSet). For any concept Cnp , we proceed to check if it is a positive (or negative) concept only if it is not a member of PE (or NE). This ensures that no concepts can occur in the concept-set more than once, so the cost of this inference algorithm in the worst case is $O(N*S)$, where N is the number of concepts in the hierarchies, and S is the upper bound on the number of suspended conditions. In the practical applications, we believe that there would not be many suspended conditions during inference, so $O(N*S)$ would be close to linear.

5. EXAMPLES

We now look at some classical problematic examples of inheritance. We show how to represent the hierarchical knowledge of these examples in CN, and how our inference algorithm works on the represented knowledge. First, consider the “elephant” example of section 3 with the following additional facts:

```
OBJECT african_elephant.
is_a: elephant.
end african_elephant.

OBJECT male_royal_elephant.
is_a: royal_elephant.
end male_royal_elephant.

OBJECT female_royal_elephant
is_a: royal_elephant.
end female_royal_elephant.
```

Now consider the instance *clyde*, which is an instance of *male_royal_elephant*, and an instance of *african_elephant*; we have $i = clyde$, $P = \{african_elephant, male_royal_elephant\}$, and $N = \{\}$. After *elephant* has been added to PE (because *elephant* is a superclass of *african_elephant*) and the condition of *is_a(elephant, grey_thing)* has been checked (because *grey_thing* is a superclass of *elephant*) we will reach the following state:

```
PE    = {elephant, african_elephant, male_royal_elephant}
NE    = {}
CSet  = {male_royal_elephant}
Susp  = {(grey_thing, royal_elephant,
         [this_frame isnot royal_elephant])}
```

We have a suspended condition here. After *royal_elephant* has been added to PE (because it is a superclass of *male_royal_elephant*), this suspended condition will be re-evaluated and will become an unsatisfied condition, so that we can determine that *grey_thing* is a negative concept of *clyde*. The final state in this example is:

```
PE    = {elephant, african_elephant,
         male_royal_elephant, royal_elephant}
NE    = {grey_thing}
CSet  = {}
Susp  = {}
```

The next example is quoted after (Padgham 1988) 6:

```
Quaker is pacifist.
Pacifist is antimilitary.
Republican is not pacifist.
Republican is football fan.
Football fan is not antimilitary.
```

There are ambiguities in these sentences. That is to say, common-sense reasoning may give a contradiction. For instance, given an instance *Nixon*, who is *quaker* and *republican*, we

may conclude from these sentences that *Nixon* is *pacifist* and is not *pacifist*. As Shastri pointed out, there are at least two distinct ways of dealing with this kind of conflict:

- 1) enumerate all the possible answers,
- 2) obtain more information to resolve such conflicts.

Etherington and Reiter's default logic approach, Padgham's lattice-based model approach, and Touretzky's formalizations¹⁰ essentially adopt the first approach to the problem of conflict. Shastri's evidential formalization tries to find the most likely solution by adding the measures of likelihood of facts to the knowledge base; the limitation is that such measures are not always available in the real world. We believe that the ambiguities result from incomplete knowledge. There must be exceptions from the hierarchical relations described in the above five sentences. The sentence "A is B" may not mean "any instance of A is an instance of B". The missing exceptions are the source of conflict.

Assume that the real meaning of these sentences is as follows:

- Only a typical quaker is pacifist.
- Pacifist is antimilitary.
- Only a typical republican is not pacifist.
- Only a typical republican is a typical football fan.
- Only a typical football fan is not antimilitary.

Suppose that every concept C has two subclasses, C_t and C_e . C_t is called a *typical C*, C_e is called an *exceptional C*. An instance i of C belongs to C_t if and only if it is a typical instance of C ; otherwise, it belongs to C_e . A typical feature of a concept is one which subjects believe applies to typical instances of the concept¹. For a concept C , we need to ask the "operator" who interacts with the representation system to list the typical features of C . We say that i is a typical instance of C if and only if it has all typical features of C . For example, suppose that an elephant's typical features are: four legs, one trunk, two big ears. An elephant that has four legs, one trunk, and two big ears is regarded as a typical elephant. An elephant that has only three legs or no trunk will not be regarded as typical.

C_e and C_t are complementary in the sense that any instance of C must belong either to C_e or C_t . That is to say, we assume that typicality and exceptionality are mutually exclusive. Complementary pairs of concepts are represented in CN by means of hierarchical slots with a *false* condition (see the frames below). Our inference algorithm automatically concludes that an instance of C_t is not an instance of C_e , and the other way around.

We denote the necessary typical and exceptional concepts in the "pacifist" example as *exceptional_republican*, *exceptional_quaker*, *exceptional_football_fan*, *typical_republican*, *typical_quaker*, and *typical_football_fan*. Only some of the frames of this example are shown below; the unshown ones can be designed similarly.

```
OBJECT quaker.
[this_frame isnot exceptional_quaker] is_a: pacifist.
end quaker.

OBJECT republican.
[this_frame is exceptional_republican] is_a: pacifist.
[this_frame isnot exceptional_republican]
  is_a: typical_football_fan.
end republican.
```

```

OBJECT exceptional_quaker.
is_a: quaker.
[false] is_a: typical_quaker.
end exceptional_quaker.

OBJECT typical_quaker.
is_a: quaker.
[false] is_a: exceptional_quaker.
end typical_quaker.

```

Given the instance *Nixon* (*quaker* but not a typical *quaker*, and *typical_republican*), the initial values of PE, NE, CSet, and Susp are

```

PE    = {exceptional_quaker, typical_republican}
NE    = {typical_quaker}
CSet  = {exceptional_quaker, typical_republican}
Susp  = {}

```

Apply the inference algorithm. When the procedure has been terminated, the values of these parameters would be

```

PE    = {exceptional_quaker, quaker,
         typical_republican, republican,
         typical_football_fan, football_fan}
NE    = {typical_quaker, pacifist, exceptional_republican,
         exceptional_football_fan, antimilitary}
CSet  = {}
Susp  = {}

```

That means *Nixon* is a quaker, a republican, a football fan, but is neither a pacifist nor an antimilitary.

In this and all the previous examples, and in general in CN, the representation is not unique. One may ask what is the method that leads to a particular representation. We can give a few general guidelines for designing a representation of hierarchies in CN:

- 1) There is a frame for every concept which has at least one superclass.
- 2) Every hierarchical relation is represented by one hierarchical slot.
- 3) If exceptions exist in a hierarchical relation, attach a condition to it.
- 4) Complementary pairs of concepts are represented by the hierarchical slots with a *false* condition.

6. CONCLUSION

The inheritance problem is dealt with in a new way in the inheritance system of Conceptual Network. Exceptions can be expressed in CN without introducing any additional mechanisms. They are explicitly represented in the conditions of the hierarchical relations; this makes monotonic inference possible. Representing the inheritance hierarchies may not be unique. So far, we can only provide a few general guidelines for designing a representation of a given problem. One of the directions of our future work is to develop a methodology of constructing a CN representation.

Our inheritance system does not include any special mechanism for resolving ambiguities, because we assume that the knowledge to be represented does not contain them; when they exist, the system will find one of the possible extensions.

REFERENCES

- [1] Dahlgren, K. (1988) *Naive Semantics for Natural Language Understanding*. Kluwer Academic Publishers.
- [2] Etherington, D. W. (1987) "Formalizing Nonmonotonic Reasoning System", *Artificial Intelligence* vol. 31, no.1, pp. 41-85.
- [3] Etherington, D. W. and R. Reiter (1983) "On Inheritance Hierarchies With Exceptions", *Proc AAAI-83*, Washington, D.C., pp. 104-108.
- [4] Fahlman, S. E. (1979) *NETL: A System for Representing and Using Real-world Knowledge*. MIT Press.
- [5] Fahlman, S. E., D. S. Touretzky and W. van Roggen (1981) "Cancellation in a Parallel Semantic Network". *Proc IJCAI-81*, pp. 257-263.
- [6] Padgham, L. (1988) "A Model and Representation for Type Information and Its Use in Reasoning with Defaults". *Proc AAAI-88*, vol. 2, pp. 409-414.
- [7] Shastri, L. (1988) *Semantic Network: An Evidential Formalization and Its Connectionist Realization*. Morgan Kaufmann.
- [8] Shastri, L. (1989) "Default Reasoning in Semantic Networks: A Formalization of Recognition and Inheritance", *Artificial Intelligence* vol. 39, no. 3, pp. 283-355.
- [9] Szpakowicz, S. (1990) "Semi-Automatic Acquisition of Conceptual Structure from Technical Texts". *Int J of Man-Machine Studies*, vol. 33, pp. 385-397.
- [10] Touretzky, D. (1986) *The Mathematics of Inheritance Systems*. Morgan Kaufman.
- [11] Touretzky, D., J. F. Horty and R. H. Thomason (1987) "A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems". *Proc IJCAI-87*, Milan, vol. 1, pp. 476-482.
- [12] Yang, L. and S. Szpakowicz (1990) "A Knowledge Representation Formalism: Conceptual Network". TR-90-27, Dept. of Computer Science, Univ. of Ottawa.
- [13] Yang, L. and S. Szpakowicz (1990) "Path Finding in Networks". *Proc SEARCC-90*.
- [14] Yang, L. and S. Szpakowicz (1991) "Planning in Conceptual Network". *Proc ICCI-91*.

COMBINING SYMBOLIC AND NUMERIC REPRESENTATIONS IN LEARNING FLEXIBLE CONCEPTS: THE FCLS SYSTEM

Jianping Zhang¹

Department of Computer Science
Utah State University
Logan, Utah 84322-4205

ABSTRACT

Many current methods of learning concepts from examples assume that concepts are precise entities, representable by a pure symbolic representation, and that concept examples are equally representative. Human concepts, however, are often flexible. They inherently lack precisely defined boundaries and have a central tendency, and their meaning is often context-dependent. Examples of these concepts are usually not all equivalent. This paper describes an approach to learning flexible concepts from examples. In this approach, a novel hybrid representation was introduced to represent flexible concepts. This hybrid representation is a combination of symbolic and numeric representations. An associated inductive learning algorithm was also presented. This approach was implemented in the Flexible Concept Learning System (FCLS) and tested on three different types of problems: the problems favorable for FCLS, the problems unfavorable for FCLS, and real world problems. The experimental results showed a strong support for the proposed flexible concept learning method.

1 INTRODUCTION

In real world applications, rare concepts are precisely defined. Instead, the meaning of concepts are often imprecise and context-dependent, these concepts are called flexible concepts [6]. Concept representations used in many learning systems, e.g. decision trees and logic-type representation, are not appropriate for describing flexible concepts. To represent flexible concepts, a representation must be capable of describing their imprecise and irregular boundary, context-dependency, central tendency and exceptions.

In the past, several representations were proposed to describe flexible concepts. These include exemplars-based representation [10][1][2] and probabilistic representations [10]. Although good results have been achieved by the systems using these representations on some domains, each of these representations has its weakness [6].

¹ This research was done, while the author was with the Artificial Intelligence Center of George Mason University. The activities of the Center are supported in part by the Defence Advanced Research Projects Agency under grant No. N00014-87-K-0874, administered by the Office of Naval Research, and in part by the Office of Naval Research under grant No. N00014-88-K-0226 and N00014-88-K-0397.

This paper presents an approach that uses a hybrid representation to describe flexible concepts. The representation is based on a simple but powerful form of *two-tiered concept representation* [5] and combines the logic and parametric representations in which both logical and parametric aspects are being adjusted in the process of learning. The method has been implemented in the system FCLS (*Flexible Concept Learning System*), and tested on a variety of problems. The problems included learning concepts with graded membership, such as congress voting, lymphatic cancer diagnosis, and n-out-m concepts, as well as concepts with sharp boundaries, such as multiplexer and DNF functions with few disjuncts. For comparison, other methods, such as C4.5 [7], were tested on the same problems. The results have shown a statistically meaningful advantage of the proposed method over the other methods, both in terms of the classification accuracy and the description simplicity. The work reported in this paper is related to Schlimmer's STAGGER [9], Utgoff's Perceptron Trees [11], Bergadano et. al's POSEIDON [3], and Salzberg's NGE. [8]

2 CONCEPT REPRESENTATION

This section introduces the hybrid concept representation used in FCLS. In this representation, a concept is described as a disjunction of extended complexes, and a similarity measure. An extended complex consists of a base complex, a set of weights, and a threshold. The similarity measure determines the degree of fit between an event and an extended complex.

2.1 BASE COMPLEX

A base complex is a disjunct represented as a complex by the attribute based Logic System VL₁ [4]. A complex in VL₁ is a conjunction of selectors. A selector is of the form:

$$[L \# R]$$

where the attribute L is called the *referee* and R is called the *referent*, which is a set of values from the domain of L. The symbol # denotes one of the relational symbols =, <, >, ≤, ≥, ≠.

2.2 WEIGHTS

Each selector of a complex is associated with a weight which reflects the degree of necessity of the selector. Its value ranges from 0 to ∞. A selector weighted as ∞ is a necessary condition of the complex, and a selector weighted as 0 is irrelevant condition. Except 0 and ∞, any other value of a weight reflects the relative importance of the selector in comparison with other selectors in the same complex.

2.3 THRESHOLD

In addition to weights, each extended complex is associated with a threshold that is a real number between 0 and 1. The threshold of an extended complex defines the boundary of the complex. An event is covered by an extended complex, if its degree of fit to the complex is larger than or equal to the threshold of the complex. Degree of fit is computed by the similarity measure. An extended complex with 1 as its threshold is equivalent to its base complex. Decreasing a threshold relaxes the requirements of the extended complex that have to be met by its instances, and generalizes the extended complex.

2.4 SIMILARITY MEASURE

The similarity measure (SM) measures the degree of fit between an event and an extended complex. The specific SM used in our current implementation maps an event from the set E and an extended complex from the set C to a real value between 0 and 1, which is the degree of fit of the event to the complex.

$$SM: E \times C \rightarrow [0..1]$$

The SM of an event e and an extended complex cpx is defined by a normalized distance measure DIS as follows:

$$SM(e, cpx) = 1 - \frac{DIS(e, cpx)}{MAXDIS(cpx)}$$

where $MAXDIS(cpx)$ is the maximum distance between events in the set E and the complex cpx . $DIS(e, cpx)$ is defined as a weighted sum of the distances between the event e and all selectors of the complex cpx :

$$DIS(e, cpx) = \sum W_i * SELDIS(e, sel_i)$$

where W_i is the weight of sel_i . $SELDIS(e, sel_i)$ is the distance between the event e and the selector sel_i and depends on the type of the variable in the selector. It is either 1 (match) or 0 (no match) for nominal variables. In case of linear variables, $SELDIS(e, sel_i)$ inversely depends on the distance of the event from the selector, normalized by dividing the largest distance between a value in the domain of the corresponding attribute and the selector.

One of the nice feature of the similarity measure is if any necessary selector of cpx is not satisfied by an event e , $SM(e, cpx) = 0$. The weight of a necessary selector is ∞ , so if the selector is not satisfied, $DIS(e, cpx) = \infty$. When $DIS(e, cpx) = \infty$, it is set to equal to $MAXDIS(cpx)$, thus $SM(e, cpx) = 0$.

2.5 CONCEPT RECOGNITION

In FCLS, an event belongs to an extended complex if the Normalized Degree of Fit (NDF) of the event to the extended complex is the largest among all extended complexes. The Normalized Degree of Fit (NDF) between an event e and an extended complexes cpx is defined as follows:

$$NDF(e, cpx) = \begin{cases} \frac{Certainty(cpx) * SM(e, cpx) - th(cpx)}{1 - th(cpx)} & SM(e, cpx) \geq th(cpx) \\ \frac{SSM(e, cpx) - th(cpx)}{th(cpx)} & SM(e, cpx) < th(cpx) \end{cases}$$

where $Certainty(cpx)$ is the certainty of cpx which is defined as the inverse of the sparseness of cpx , $th(cpx)$ is the threshold of cpx .

2.6 EXAMPLES

To illustrate the idea of the hybrid representation, let us consider a simple imaginary concept "R-ball". The meaning of the concept R-ball is defined as three disjuncts:

- (SHAPE = round) & (BOUNCES = yes) or
- (SHAPE = round) & (SIZE = medium v large) or
- (BOUNCES = yes) & (SIZE = medium v large)

By using the hybrid representation, these three disjuncts merge into one extended complex:
[SHAPE = round : 1] & [BOUNCES = yes : 1] & [SIZE = medium v large : 1]

$$\text{Threshold} = \frac{2}{3} = 0.67$$

The number following ':' is the weight of the selector. The base complex:

[SHAPE = round] & [BOUNCES = yes] & [SIZE = medium v large]

represents the central tendency of the concept R-ball, all of the three selectors are equally important. The meaning defined by the extended complex is that an object that satisfies any two or more of the three selectors is a R-ball, otherwise it is not a R-ball. Furthermore, it tells that balls that satisfy all of the three selectors are typical R-balls, while those which only satisfy two of the three selectors are less typical.

Now suppose the meaning of the concept R-ball changes a little, and all R-balls must be round. The new meaning of the concept R-ball is defined by two disjuncts:

(SHAPE = round) & (BOUNCES = yes) or

(SHAPE = round) & (SIZE = medium v large)

These two disjuncts are combined into one extended complex:

[SHAPE = round : ∞] & [BOUNCES = yes : 1] & [SIZE = medium v large : 1]

$$\text{Threshold} = \frac{1}{2} = 0.5$$

In this extended complex, the selector [SHAPE = round] is necessary, and must be satisfied by all R-balls. The other two selectors are not necessary, and one of them must be satisfied by a R-ball.

3 THE LEARNING ALGORITHM

Table 1 defines the learning algorithm which works in an iterative fashion. In each iteration, the concept whose description has the largest error omission is generalized by generating a new acceptable extended complex to minimize the error omission of the concept. FCLS provides users with two parameters: *MAX-ERR-RATE* and *MIN-COVERAGE*. These two parameters are used as thresholds. An extended complex is acceptable if

$$(1) \frac{p}{p_{\text{pos}}} \geq \text{MIN-COVERAGE}, \text{ and } (2) \frac{n}{p + n} \leq \text{MAX-ERR-RATE}.$$

The error omission of a concept description is the percentage of the number of the positive examples that are not covered by the description. If the fraction of correctly classified examples are larger than *MAX-ERR-RATE*, the algorithm terminates and outputs the current descriptions, otherwise it repeats.

Let DES be empty

Repeat

 Select the concept CNPT that has the largest error omission.

 Generalize CNPT by generating an acceptable extended complex CPX

 Add CPX and EXE into DES

Until error-rate(DES) < *MAX-ERR-RATE*

Return DES

Table 1. The Learning Algorithm in FCLS

3.1 THE COMPLEX GENERATION ALGORITHM

The complex generation algorithm generates the extended complex for a given concept from a set of positive and negative examples. The process of generating the extended complex is divided into two phases. The first phase generates a set of base complexes that satisfy the consistency requirement specified by the parameter *MAX-ERR-RATE*. The base complexes generated in the first phase are optimized in the second phase. Before describing the two algorithms in the two phases, we first introduce some terminology used in the algorithms. Let us suppose e is an example, and cpx is an extended complex. e is called strictly covered example, if $SM(e, cpx) = 1$, that is e satisfies all conditions of cpx . e is flexibly covered by cpx , if $SM(e, cpx) \geq th(cpx)$. e is nearly covered by cpx , if $th(cpx) > SM(e, cpx) \geq pth(cpx)$. Where $th(cpx)$ is the threshold of cpx , $pth(cpx)$ is the potential threshold of cpx which is less than $th(cpx)$ and used to decide nearly covered examples.

3.1.1 PHASE 1: THE BASE COMPLEXES GENERATION ALGORITHM

The algorithm generates a set of the most general base complexes that satisfy the consistency requirement. Table 2 specifies the algorithm. It starts with the most general base complex which strictly covers the whole instance space. In order to find base complexes that satisfy the consistency requirement, the strictly covered negative examples must be excluded. The technique used in the algorithm is similar to the *star* algorithm of AQ [4] that performs a beam search. During each cycle, the consistency of each base complex in STAR is tested. If the consistency is high enough, the base complex is added to the set of CONSISTENT-CPXES and removed from STAR. Otherwise, the base complex is specialized by removing a value from one of its selectors. This specialization is repeated for each of all selectors of the complex. The value removed from a selector is chosen to maximize the number of negative examples and minimize the number of positive examples excluded from the base complex. This yields several new base complexes, each of which covers fewer negative examples. The new star is the union of these newly specialized base complexes. A certain maximum number (*MAXSTAR*) of these base complexes are selected for further processing. This set of base complexes is selected based on their potential quality. When STAR is empty, the algorithm terminates with a set of base complexes whose inconsistency (error rate) is smaller than *MAX-ERR-RATE*. Figure 1(a) shows the most general base complex that the algorithm starts with and Figure 1(b) shows the set of consistent base complexes that the algorithm ends up with.

```

Let STAR be the set containing the most general complex that covers all events.
Let CONSISTENT-CPXES be empty.
Repeat
  Let NEWSTAR be empty
  For each complex CPX in STAR
    For each attribute
      select a value to remove from CPX so that a more specific complex NEWCPX is generated.
      if error-rate(NEWCPX) ≤ MAX-ERR-RATE ,
        then add NEWCPX into CONSISTENT-CPXES
        else add NEWCPX into NEWSTAR
  Let STAR be MAXSTAR complexes with the largest potential quality in NEWSTAR.
until STAR is empty
Return CONSISTENT-CPXES

```

Table 2: The Base Complex Generation Algorithm

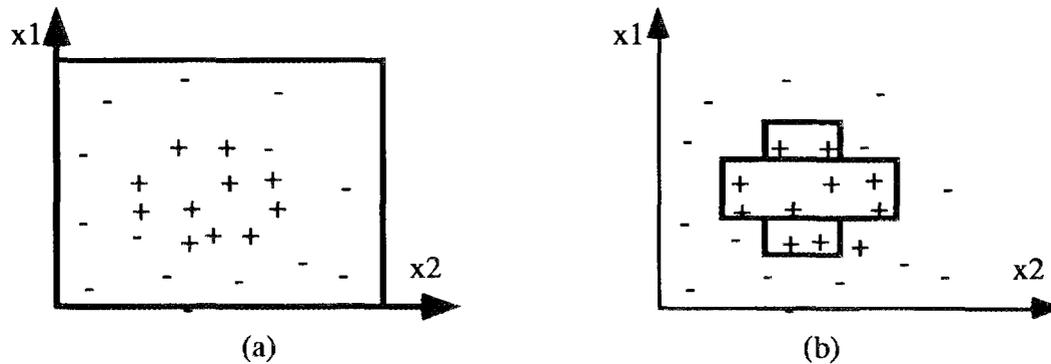


Figure 1: An illustration of the function of the phase 1

3.1.2 PHASE 2: THE EXTENDED COMPLEX OPTIMIZATION ALGORITHM

The extended complex optimization algorithm optimizes the complexes generated in phase 1 by decreasing the thresholds of the complexes so that more positive examples can be covered. In order to decrease the threshold of an extended complex without increasing inconsistency, the degree of fit of nearly covered negative examples must be reduced so that the threshold can be decreased without covering more negative examples. The way to reduce the degree of fit of nearly covered negative examples is to specialize the base complex by removing some values of selectors that occur on many nearly covered negative examples and few nearly covered positive examples. The algorithm also performs a general-to-specific beam search. In this algorithm, the threshold is adjusted (often decreased) while the base complex is specialized. Thus, an extended complex is often generalized although its base complex is specialized.

Table 3 specifies the extended complex optimization algorithm. The algorithm first transfers the base complexes generated in phase 1 by computing its new weights and new threshold. The weight learning algorithm will be introduced in section 3.2. The threshold is determined so that the 'best' quality of the complex is achieved. The STAR is initialized as *MAXSTAR* of these complexes with the highest potential quality. Then the 'best' acceptable extended complex is selected from the set of optimized initial extended complexes as the initial 'best' complex BEST-CPX. This 'best' extended complex is subject to replacement by a better extended complex during the process of optimization. After the algorithm terminates, BEST-CPX is output. The 'best' extended complex is the complex with the highest quality. If no acceptable extended complex can be generated, BEST-CPX is empty when the algorithm terminates.

The algorithm repeats the beam search until the stop condition is satisfied. In each cycle of the loop, a set of new extended complexes is generated. The quality and potential quality of each newly generated extended complex are evaluated respectively. The acceptable complex with the highest quality replaces the complex in BEST-CPX, if its quality is larger than or equal to the quality of the complex in BEST-CPX. The complexes with low potential quality are removed from NEWSTAR. Issues about quality and the potential quality were discussed in [12]. The *MAXSTAR* new extended complexes with the highest potential quality are selected for further improvement. *MAX-TRIES* is an integer parameter which controls the execution of the loop. If BEST-CPX has not been improved in *MAX-TRIES* steps, the algorithm stops.

```

For each complex CPX in CONSISTENT-CPXES (generated in Phase 1)
  Compute the weights and threshold for CPX
Let STAR be MAXSTAR complexes with the highest potential quality in CONSISTENT-CPXES
If there exist some acceptable complexes in CONSISTENT-CPXES
  then let BEST-CPX be the acceptable complex with the highest quality
  else let BEST-CPX be empty
Let NO-IMPROVEMENT be 0
Repeat
  Let NEWSTAR be empty
  For each complex CPX in STAR
    For each attribute
      select a value to remove from CPX to generate a new complex NEWCPX
      compute the weights and threshold for NEWCPX
      if NEWCPX is acceptable and has equal or higher quality than BEST-CPX
        then replace BEST-CPX by NEWCPX
        set NO-IMPROVEMENT to 0
      else add 1 to NO-IMPROVEMENT
    add NEWCPX into NEWSTAR
  Remove all complexes that cannot be improved from NEWSTAR
  Let STAR be MAXSTAR complexes with the largest potential quality in NEWSTAR.
until NO-IMPROVEMENT > MAX-TRIES or STAR is empty
Return BEST-CPX

```

Table 3: The Extended Complex optimization algorithm

Fig. 2(a) shows the initial extended complexes that the algorithm starts with and Fig. 2(b) shows the extended complex that the algorithm ends up with. In Fig. 2(b), the circle represents an extended complex, and the square inside the circle is its base complex. It can be seen that the base complex in Fig. 2(b) is more specific than the two base complexes in Fig. 2(a), but the extended complex is more general than both of the two base complexes.

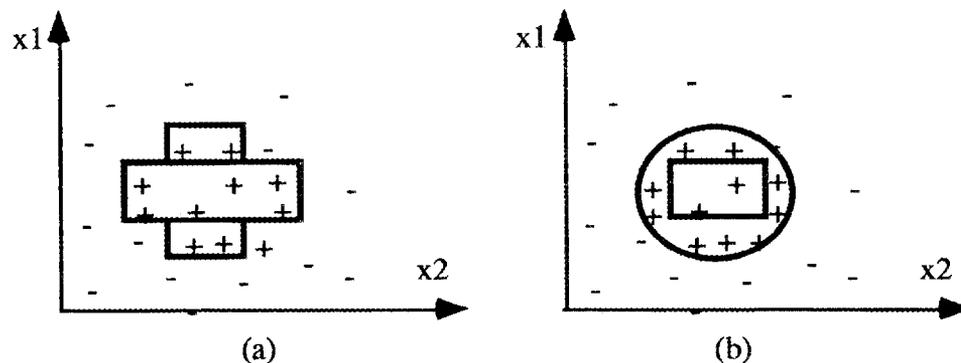


Figure 2: An illustration of the function the phase 2

3.2 WEIGHT LEARNING ALGORITHM

In the hybrid representation, each selector of an extended complex is associated with a weight which is the degree of necessity of the selector. A weight is a real value ranging from 0 to $+\infty$. The larger a weight of a selector, the more necessary the selector. The weights of an extended complex are computed during learning. In computing the weight of a selector, the algorithm counts the number of positive and negative examples that do not

match the selector. In the method, the weight of the selector SEL $w(\text{SEL})$ is computed as follows:

$$w(\text{SEL}) = \frac{p(\text{unmatched} \mid \text{NEG})}{p(\text{unmatched} \mid \text{POS})}$$

where $p(\text{unmatched} \mid \text{NEG})$ and $p(\text{unmatched} \mid \text{POS})$ are the fraction of positive and negative examples which do not match with SEL. $w(\text{SEL})$ ranges from 0 to $+\infty$. When the selector SEL is satisfied by all positive examples, $p(\text{unmatched} \mid \text{POS}) = 0$ so that $w(\text{SEL}) = +\infty$ and the selector SEL is necessary. When the selector SEL is satisfied by all negative examples, $p(\text{unmatched} \mid \text{NEG}) = 0$ so that $w(\text{SEL}) = 0$ and the selector is totally unnecessary. This case occurs seldom, because such a selector is usually removed in the process of complex generation. The fewer negative examples satisfy the selector SEL, the larger $p(\text{unmatched} \mid \text{NEG})$ and $w(\text{SEL})$. The more positive examples satisfy the selector SEL, the smaller $p(\text{unmatched} \mid \text{POS})$, therefore the larger $w(\text{SEL})$.

4 EXPERIMENTS WITH FCLS

To evaluate the approach described in this paper, a number of experiments were conducted on various domains with FCLS. This section first outlines the experimental methods and the domains, then reports the details of the experimental results.

4.1 EXPERIMENTAL DESIGN

To thoroughly test FCLS, six artificial domains, three favorable to FCLS and three unfavorable, were selected for the experiments. [12] also described experiments from two real world domains. Three learning methods, the base-cpx, the no-weight, and the c-weight, were involved in all experiments. The base-cpx method generates a disjunction of base complexes as a concept description that is equivalent to a DNF expression. The base-cpx method provides the performance baseline for other methods. The no-weight method generates extended complexes with threshold adjusting only, no weight learning is involved. The c-weight method generates an extended complex with both threshold adjusting and weight learning. In addition to these three methods, the decision tree learning system C4.5 [7] was run on the same domains with pruning. The performance of FCLS was evaluated on classification accuracy and description complexity. Classification accuracy was measured as the percentage of correct classifications made by the concept description on a set of 1000 test events. Description complexity was measured by the number of extended complexes involved in a description. The complexity of decision trees is measured by the number of leaves in a tree. In all experiments, FCLS was run on randomly generated training sets of various sizes: 100, 200, 300, and 400 examples. For each training set size, FCLS was run on four different randomly generated training sets. The results reported in Figure 3 and 4 are the average of the four runs. The results accompanied with a 95% confidence interval calculated using a Student t-test were reported in [12].

4.2 EXPERIMENTS ON THE DOMAINS FAVORABLE TO FCLS

The experiments described in this section were performed on three specially designed domains, called designed domain I to III. These domains were specially designed to test the novel features of the hybrid representation and the associated learning algorithm in FCLS. **Designed Domain I** contains two classes, positive and negative, and 10 nominal

attributes each of which has four values: 0, 1, 2, and 3. The rule for distinguishing positive class from negative class has the general form of "at least k of n conditions are satisfied." Specifically, the rule is "if the values of any 5 or more of the first 7 attributes of an event are equal to 0 or 1, then the event belongs to positive class, otherwise it belongs negative class".

Designed Domain II consists two classes, positive and negative. Eight linear attributes are involved in this domain. The domains of the eight linear attributes are same and include four values 0, 1, 2 and 3. The positive class is described by six conditions, two of which are as twice important as the other four conditions. Specifically, the positive class is expressed by one extended complex:

$$[x_1 = 0 \vee 1]:2 \ \& \ [x_2 = 0 \vee 1]:2 \ \& \ [x_3 = 0 \vee 1]:1 \ \& \ [x_4 = 0 \vee 1]:1 \ \& \ [x_5 = 0 \vee 1]:1 \ \& \ [x_6 = 0 \vee 1]:1$$

Threshold = $5/8 = 0.625$

Designed Domain III contains 15 nominal binary attributes, and two classes: positive and negative. The events of the positive class are described by two extended complexes, each of which consists of 6 selectors, two of which are as twice important as the other four. The positive class is described by the disjunction of the following two extended complexes:

Complex 1:

$$[x_1 = 0]:2 \ \& \ [x_2 = 0]:2 \ \& \ [x_3 = 0]:1$$

$$[x_4 = 0]:1 \ \& \ [x_5 = 0]:1 \ \& \ [x_6 = 0]:1$$

Threshold = $5/8 = 0.625$

Complex 2:

$$[x_7 = 0]:2 \ \& \ [x_8 = 0]:2 \ \& \ [x_9 = 0]:1$$

$$[x_{10} = 0]:1 \ \& \ [x_{11} = 0]:1 \ \& \ [x_{12} = 0]:1$$

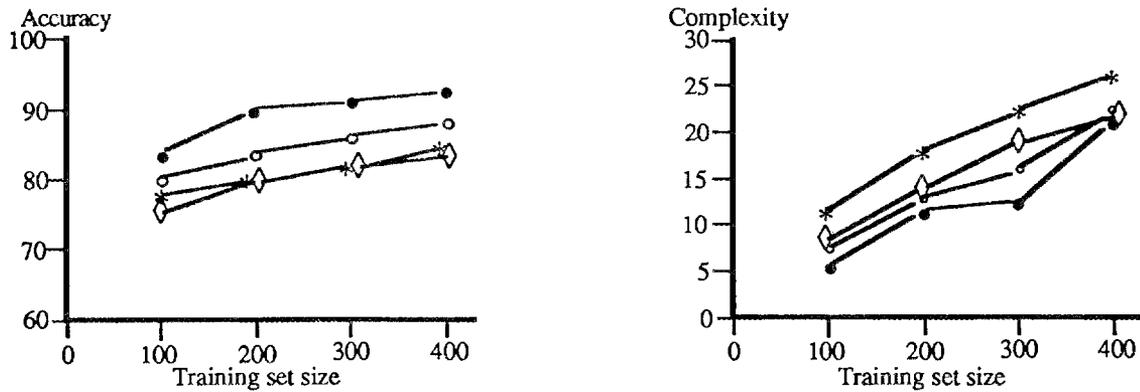
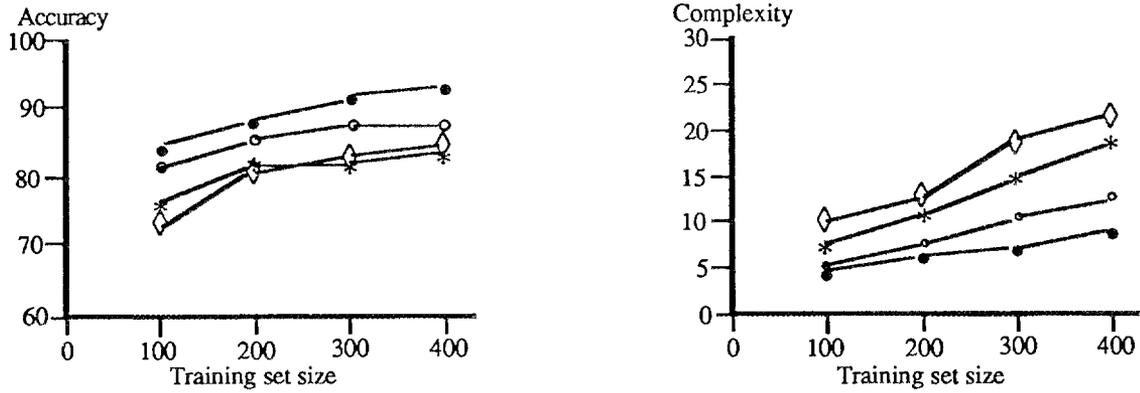
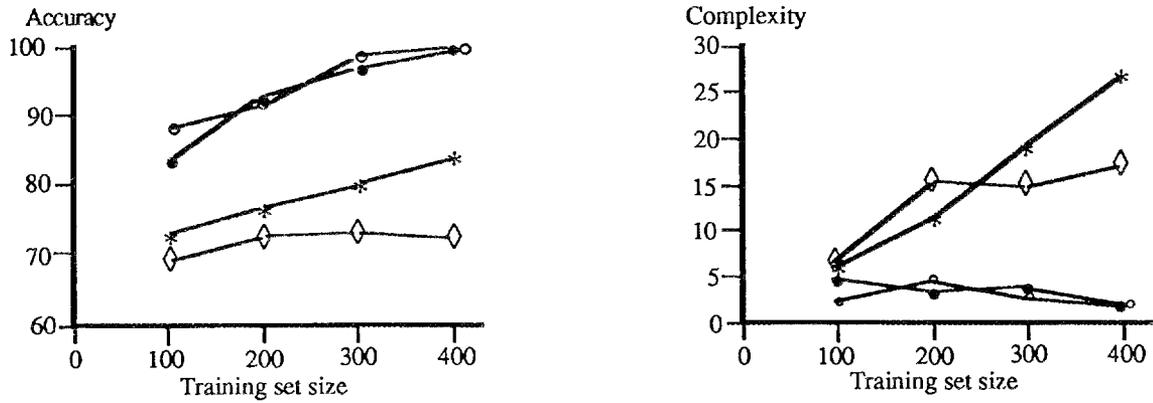
Threshold = $5/8 = 0.625$

Figure 3 shows the results of the experiments from the three favorable domains. In all three domains, improvements were achieved on both accuracy and complexity by the no-weight and the c-weight methods over the base-cpx method and C4.5 at all training set sizes. A significant improvement was achieved in the Designed Domain I. The results from Designed Domain I show that the no-weight and c-weight methods have very similar performance. This is because all conditions of the target concept description are equally important, and weights play no role. The c-weight method outperformed the no-weight method in Designed Domain II and Designed Domain III. These improvements are due to the weight learning. In these two domains, selectors in extended complexes are weighted differently.

4.3 EXPERIMENTS ON THE DOMAINS UNFAVORABLE TO FCLS

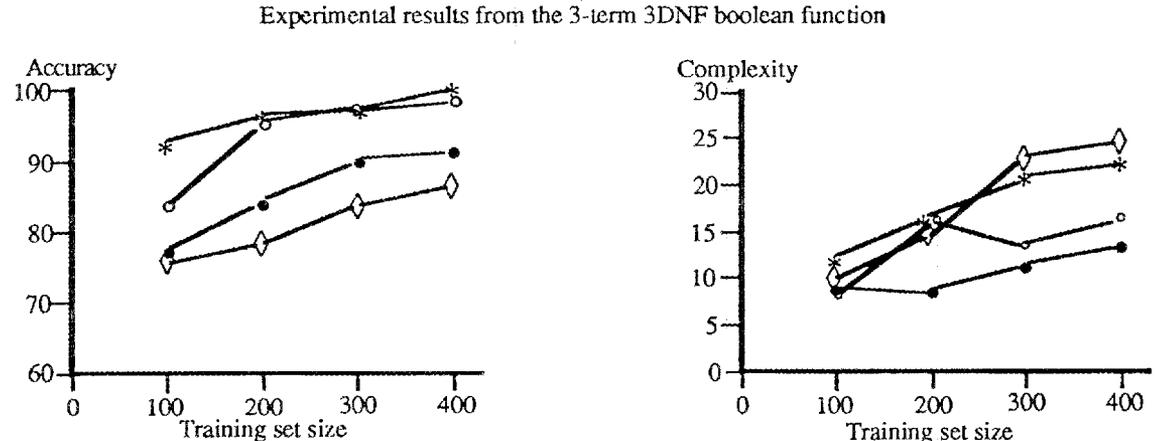
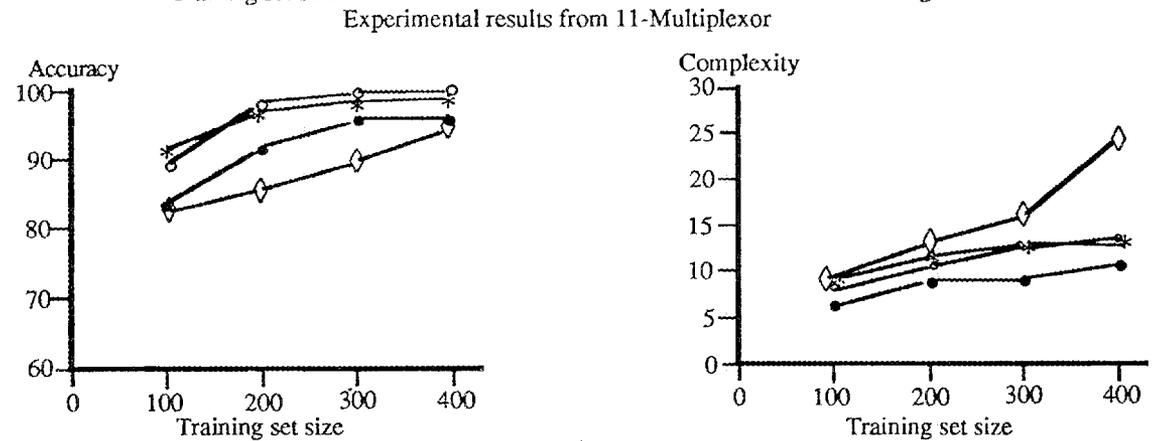
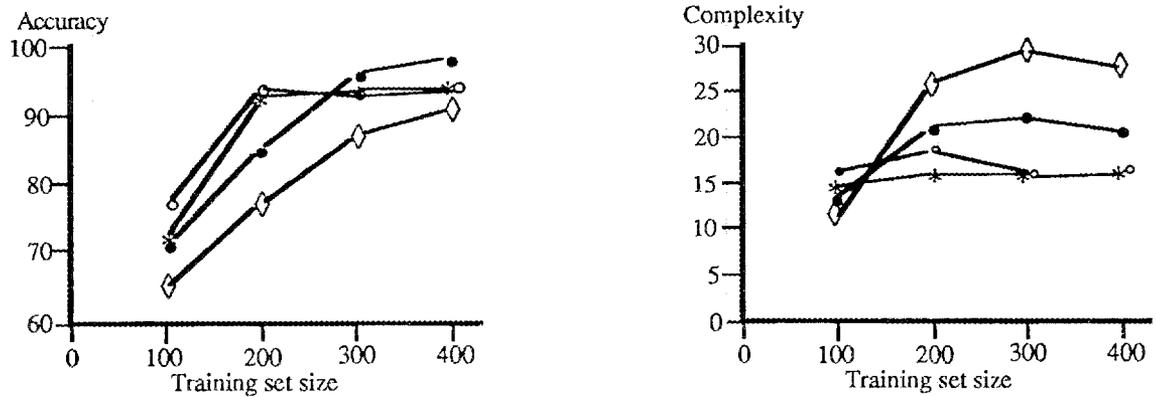
This section describes the experiments from three unfavorable domains: 11-multiplexor, 3-term 3DNF and 4-term 3DNF. The hybrid representation has no advantage over logic type representations in representing the concepts involved in these domains. Adversely, the hybrid representation increases difficulties to learn these concepts because of the less representational bias enforced by the representation.

The results from the three unfavorable domains are reported in Figure 4. Except in 11-multiplexor at size 300 and 400, the accuracy of the c-weight method is worse than that of the base-cpx and no-weight methods, especially at small training sizes. This result is due to the weak representational bias enforced by the hybrid representation. In spite of the problem, the accuracy of the methods with weight learning is still comparable with the accuracy of C4.5. Except in the domain of 11-multiplexor, the c-weight generated simpler descriptions. One important and interesting result is that the accuracy obtained through the no-weight method is similar to the accuracy of the base-cpx method in all three domains, in some experiments, the accuracy of the no-weight method is even slightly better than the base-cpx. In fact, in many experiments, the base-cpx method and the no-weight method generated the exactly same descriptions. This interesting result shows that the no-weight method works very well in adjusting the representation for a given problem, but the weight learning methods does not.



• The c-weight method ◊ The no-weight method * The base-cpx method ◊ C4.

Figure 3: Experimental Results from Favorable Domains



● The c-weight method ○ The no-weight method * The base-cpx method ◇ C4.

Figure 4: Experimental Results from Unfavorable Domains

5 CONCLUSION AND FUTURE WORK

This paper described an novel approach to learning flexible concepts. In this approach, a hybrid representation that combines symbolic and numeric representations was proposed to explicitly describe central tendencies of flexible concepts and extend the meaning of concepts by a threshold and a similarity measure. An associated algorithm was designed and implemented to automatically acquire both symbolic and numeric descriptions. The experimental results are very promising and encouraging.

A number of problems need to be addressed in the future. First, FCLS should be augmented with a knowledge based semantic similarity measure. Second, an incremental version of the approach needs to be designed. Third, a better weight learning algorithm should be studied. Finally, the method of constructive induction will be incorporated into FCLS.

REFERENCES

1. Aha, D., Kibler, D., and Albert, M., "Instance-Based Learning Algorithms", *Machine Learning* 6, (1991)
2. Bareiss, E.R., Porter, B.W., and Craig, C.W., "Protos: An Exemplar-based Learning Apprentice," *Machine Learning: An Artificial Intelligence Approach V III*, 63-111 (1990).
3. Bergadano, F., Matwin, S., Michalski, R.S., and Zhang, J., "Learning Two-tiered Descriptions of Flexible Concepts," accepted for publication in *Machine Learning*.
4. Michalski, R.S., "A Theory and Methodology of Inductive Learning." In *Machine Learning: An Artificial Intelligence Approach*, 83-134 (1983).
5. Michalski, R. S., "How to Learn Imprecise concept: A Method Employing a Two-Tiered Representation for Learning", Proceedings. of the Fourth International Workshop on Machine Learning, Irvine, CA, pp. 50-58, (1987).
6. Michalski, R.S., "Learning Flexible Concepts: Fundamental Ideas and a Method Bases on Two-Tiered Representation," *Machine Learning: An Artificial Intelligence Approach V III*, 63-111 (1990).
7. Quinlan, J. R., "Simplifying decision trees." In *International Journal of Man-Machine Studies*, vol. 27, (1987).
8. Salzberg, S., "A Nearest Hyperrectangle Learning Method," *Machine Learning* 6, (1991)
9. Schlimmer, J. C., *Concept Acquisition Through Representational Adjustment*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, (1987).
10. Smith, E.E. and Medin, D.L., *Categories and Concepts*, Harvard University Press, Cambridge, MA, (1981)
11. Utgoff, P. E., "Perceptron Trees: A case study in hybrid concept representations." In *Proceedings of the seventh National Conference on Artificial Intelligence*, (1988).
12. Zhang, J., "Learning Flexible Concepts from Examples: Employing the Ideas of Two-Tiered Concept Representation." Ph.D Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, (1990).

INTERNAL DISTRIBUTION

- | | |
|--------------------|------------------------------|
| 1. B. R. Appleton | 28. D. B. Reister |
| 2. J. E. Baker | 29. J. C. Schryver |
| 3. A. L. Bangs | 30. P. F. Spelt |
| 4. M. Beckerman | 31. F. J. Sweeney |
| 5. R. J. Carter | 32. M. A. Unseren |
| 6. J. R. Einstein | 33. R. C. Ward |
| 7. C. W. Glover | 34-35. Laboratory Records |
| 8-12. K. S. Harber | Department |
| 13. J. P. Jones | 36. Laboratory Records, |
| 14. H. E. Knee | ORNL-RC |
| 15. G. Liepins | 37. Document Reference |
| 16-20. R. C. Mann | Section |
| 21. E. M. Oblow | 38. Central Research Library |
| 22-26. F. G. Pin | 39. ORNL Patent Section |
| 27. S. A. Raby | |

EXTERNAL DISTRIBUTION

40. Dr. Peter Allen, Department of Computer Science, 450 Computer Science, Columbia University, New York, NY 10027
41. Dr. Wayne Book, Department of Mechanical Engineering, J. S. Coon Building, Room 306, Georgia Institute of Technology, Atlanta, GA 30332
42. Professor Roger W. Brockett, Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138
43. Professor John J. Dorning, Department of Nuclear Engineering and Physics, Thornton Hall, McCormick Rd., University of Virginia, Charlottesville, VA 22901
44. Dr. Steven Dubowsky, Massachusetts Institute of Technology, Building 3, Room 469A, 77 Massachusetts Ave., Cambridge, MA 02139
45. Dr. Avi Kak, Department of Electrical Engineering, Purdue University, Northwestern Ave., Engineering Mall, Lafayette, IN 47907
46. Dr. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815
47. Dr. Oscar P. Manley, Division of Engineering, Mathematical, and Geosciences, Office of Basic Energy Sciences, ER-15, U.S. Department of Energy - Germantown, Washington, DC 20545
48. Professor Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green St., Urbana, IL 61801
- 49-178. Dr. Zbigniew Ras, University of North Carolina at Charlotte, Department of Computer Science, Kennedy Bldg., Charlotte, NC 28223
179. Dr. Wes Snyder, Department of Radiology, Bowman Gray School of Medicine, 300 S. Hawthorne Dr., Winston-Salem, NC 27103
180. Professor Mary F. Wheeler, Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77251
181. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831-8600
- 182-191. Office of Scientific Technical Information, P.O. Box 62, Oak Ridge, TN 37831