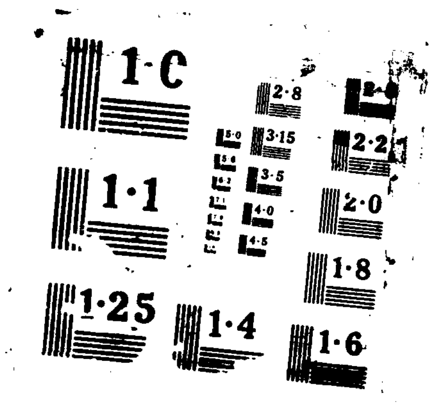
AD-A193 107 DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS B CHANDRASEKARAN 08 APR 88 1/2

UNCLASSIFIED AFOSR-TR-88-0572 AFOSR-87-0090 F/G 12/9 NL


1·0
2·8
2·5
5·0
3·15
2·2
5·6
3·5
1·1
4·0
2·0
4·5
1·8
1·25
1·4
1·6

2

AFOSR-TR- 88-0572

RF Project 765799/719026
Technical Reports & Reprints

DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL

B. Chandrasekaran
Department of Computer and Information Science

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Bolling AFB, D.C. 20332

Grant No. AFOSR-87-0090

April 1988

OSU **The Ohio State University Research Foundation**
1314 Kinnear Road
Columbus, Ohio 43212

88 - 16 106

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 765799/719026	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-0572

6a. NAME OF PERFORMING ORGANIZATION The Ohio State University Office of Sponsored Programs	6b. OFFICE SYMBOL (If applicable) OSURF	7a. NAME OF MONITORING ORGANIZATION AFOSR/NM Bolling AFB DC 20332-6448
6c. ADDRESS (City, State, and ZIP Code) 1314 Kinnear Road Columbus, Ohio 43212		7b. ADDRESS (City, State, and ZIP Code) AFOSR/NM Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Office of Scientific Research	8b. OFFICE SYMBOL (If applicable) AFOSR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-87-0090
8c. ADDRESS (City, State, and ZIP Code) BK1 410 Bolling Air Froce Base Washington, D.C. 20332		10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102F	2304	A7	

11. TITLE (Include Security Classification)
Distributed Knowledge Base Systems for Diagnosis and Information Retrieval - Unclassified

12. PERSONAL AUTHOR(S)
B. Chandrasekaran

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ______ TO ______	88/4/8	152

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes work done in artificial intelligence. Research is reported on in the following areas: applications of generic task methodology to learning, design, and planning; theoretical and critical studies of connectionism, neural nets, and symbolic computation; and studies of the uses of distributed systems for concurrent algorithms/implementations for abductive assembly, a basic generic problem solving task.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Abe Waksman	22b. TELEPHONE (Include Area Code) (202) 767-5025	22c. OFFICE SYMBOL AFOSR/NM

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

OSU The Ohio State University

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
2036 Neil Avenue Mall
Columbus, Ohio 43210-1277
Phone 614-422-0923

April 8, 1988

Air Force Office of Scientific Research
Building 410
Bolling AFB,
Washington, D.C. 20332
Attn: AFOFSR/NM Abe Waksman

Dear Dr. Waksman:

Enclosed are nine articles and technical reports that have been supported in part by AFOFSR Grants 87-0090 and 82-0255. Research is reported on:

Applications of generic task methodology to learning, design, and planning.

Theoretical and critical studies of connectionism, neural nets, and symbolic computation.

Studies of the uses of distributed systems for concurrent algorithms/implemenations for abductive assembly, a basic generic problem solving task.

If you have any questions please contact either Professor Chandrasekaran or me at 614-292-8578.

Sincerely,

Dale Moberg

Dr. Dale Moberg
Assistant Director

Enclosures:

Tom Bylander and Michael A. Weintraub,
A Corrective Learning Procedure Using Different Explanatory Types

Douglas R. Myers, James F. Davis, David J. Herman,
A Task Oriented Approach to Knowledge-Based Systems
for Process Engineering Design

Donald Sylvan, Ashok Goel, B. Chandrasekaran,
An Information Processing Model of Japanese Foreign and
Energy Policy Decision Making

Ashok Goel, B. Chandrasekaran, Understanding Device Feedback

B. Chandrasekaran, Ashok Goel, Dean Allemang,
Connectionism and IP Abstractions:
The Message Still Counts More than the Medium

John Kolen, Ashok Goel, Dean Allemang
On Some Experiments With Neural Nets

B. Chandrasekaran, Ashok Goel,
From Numbers to Symbols to Knowledge Structures:
AI Perspectives on the Classification Task

Ashok Goel, P. Sadayappan, N. Soundarajan,
Distributed Synthesis of Composite Explanatory Hypotheses

Ashok Goel, P. Sadayappan, John R. Josephson,
Distributed Synthesis of Composite Hypotheses,

A Corrective Learning Procedure Using Different Explanatory Types[1]

Tom Bylander and Michael A. Weintraub
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

Corrective learning is the alteration of a system's existing knowledge structures to produce the correct answer when the system's existing structures fail by producing an incorrect response. An explanation-based solution is to compare explanations of why the system produced its incorrect answer with explanations of the correct answer. Explaining the system's answer would be trivial if a single production rule concluded the answer directly from the data. However, the answers from the system we are building will have uncertainty, and credit assignment will involve larger knowledge structures. The problem we are working on is to see how different problem solving structures and underlying models -- and the different types of explanations coming from each -- affect the learning process in the context of corrective learning.

Our work differs from most EBL approaches in the nature of the explanations the system will be producing and using. The usual explanation-based approach is achieved by the construction of a proof showing how an example is an element of some class. The proof can be used to generate a list of sufficient conditions for the identification of some concept. The explanations our work involves can not be construed in the same manner. The answers our system will generate allow for certain conclusions to be inferred from the data, but these conclusions are probabilistic in nature and not definitive. As a result, our system will not produce exact proofs about how some instance belongs to a concept. Instead, our system will only be able to identify a probabilistic relationship between a set of conditions and a concept.

The particular domain we are working in is pathologic gait analysis. Gait analysis is non-trivial. The problem is to properly diagnose which muscles and joints are causing deviations in the gait cycle. For example, patients with cerebral palsy, a disease affecting motor control, typically have several muscles that function improperly in different phases of the gait cycle. The malfunctions in the case of cerebral palsy are improper contractions of the muscles -- both in terms of the magnitude and timing of the muscles -- during the phases of the gait cycle. The problem of diagnosing which muscles and joints are at fault is complicated by interactions between limb segments and attempted compensations by other muscles. In addition, many internal parameters cannot be directly or even indirectly measured using current technology. For example, EMG data is at best a qualitative measure of muscle forces [Simon82].

To perform diagnosis for this kind of problem, our system will consist of structured diagnostic knowledge and a qualitative physical model of human walking. The input to the diagnostic system is the information gathered about a patient by the Gait Analysis Laboratory at the Ohio State University. The data is of three types: clinical, historical, and motion. Clinical data is the result of a physical examination of the patient, and identifies the range of motion of joints by several physical tests. EMG information,

[1]This research is supported by grants 82048-02 from the National Institute for Handicapped Research and 87-0090 from the Air Force Office of Scientific Research.

identifying muscle activity, is also collected. An EMG is not collected on every muscle because of the difficulty involved in attaching electrodes to certain muscle groups. Historical data includes information about any past medical procedures or diagnoses. Motion data identifies the angular position of the patient's joints during the different gait phases. This information is recorded for each plane of interest. The output from the diagnostic system will be an explanation of the malfunctioning gait components so an appropriate therapy can be prescribed. The problem solving structures we are using are based on the theory of generic tasks [Chandra86]. The particular generic tasks involved in the system are abductive assembly, hierarchical classification, and hypothesis matching, respectively used for constructing composite malfunction hypotheses, selecting plausible malfunctions, and combining evidence for and against malfunctions.

In [Chandra87], three types of explanation are identified with knowledge-based systems. These are: (1) trace of run-time, data-dependent, problem solving behavior, (2) understanding the control strategy used by the program in a particular situation, and (3) justifying a piece of knowledge by how it relates to the domain. In our system, the first two types of explanation will be produced by compiled diagnostic knowledge.

To show how the first two explanation types arise, consider the generic task of hierarchical classification. To perform diagnostic reasoning, nodes in a classification hierarchy can be used to represent general and specific malfunctions. During problem solving, the nodes are activated in a top-down fashion and determine their applicability to the current case. Each malfunction that is considered is evaluated by compiled knowledge that matches its features against the data. The confidence value of a malfunction in the classification hierarchy is linked to the data that produced it. This is a type 1 explanation. An example of a type 2 explanation would be to describe why a malfunction was or was not considered. For example, if the confidence value of a general malfunction is low, more specific malfunctions might not be considered.

Type 3 explanations will be produced by the qualitative physical model. These explanations will point out the atypical data that a suspected malfunction would explain, i.e., if the malfunction were true, then the malfunction would be considered the cause of the data. In our system, the qualitative physical model is being implemented by qualitative differential equations [deKleer84, Kuipers86], which will be used to determine how various influences such as muscles and body weight give rise to the observed motion. The model will not be sufficient to identify the correct diagnosis because each part of the observed motion has several possible causes and because of the inherent ambiguity of qualitative models.

The learning in the system will be fault driven, i.e., an incorrect diagnosis is used to focus the learning process. The system, already possessing knowledge about the domain, albeit imperfect, gives an answer to be verified by the domain expert. If the answer is deemed incorrect, the expert provides the "correct" answer. The system must identify how the original answer differs from the correct answer and infer why the expert's answer is better. The system must identify which parts of the problem solving structure caused the incorrect solution, and then modify the structure appropriately.

Explanation of generic task structures (types 1 and 2) will be used to determine which knowledge structures might be at fault. Explanation of the qualitative physical model (type 3) will be compared to the type 1 explanation to select the faulty structure, which might be decomposable into several smaller knowledge structures to be further searched. Once the specific location of an error is found, the type 3 explanation specifies what data should have been used and the other explanation types specify what kinds of adjustments in confidence values will result in preferring the correct answer over the incorrect answer. The following procedure outlines our approach to this problem[2]:

1. Identify initial differences between the system's diagnosis and the correct diagnosis. These differences indicate the sections of the problem solving which need reconsideration. Each difference provides a point from which to focus the learning process. These differences implicate not only the actual compiled knowledge structure that produced the bad judgment, but also the set of decisions leading to the judgment.
2. For each difference, generate explanations of why the system reached its judgment. Specifically, identify the data used in support of the bad judgment (type 1 explanations), and identify the set of decisions leading to the judgment in question (type 2 explanations).
3. Find any commonalities between the explanations of the system's incorrect judgments. Having identified how the incorrect judgment was produced, find any common search strategy or data analysis used in judgments resulting in the set of differences. This step involves comparing the type 2 and 1 explanations produced for each difference, and finding the intersection.
4. Sort the set of commonalities and bad judgments in order of degree of potential effect on correcting the answer if modified, e.g., if a common decison underlies two incorrect judgments, then the changing the common decision may correct both problems.
5. Check consistency. For each element in the set of commonalities and bad judgments, compare the type 3 explanation produced by the qualitative model to the type 1 explanation of the judgment. (The qualitative model does not model the system's control structures, so it does not make sense to include type 2 explanations in this comparison.)
6. Inconsistencies found in the type 1 explanation identify points to correct. Such inconsistencies include: not using all causally relevant information, using data with no causal connection, the sensitivity of some information for decision making is overrated/underrated, etc.
7. Suggest modifications to overcome the inconsistencies. Generate alternatives to the incorrect judgments consistent with the type 3 explanations. This step will focus on making as few changes as possible to correct the overall answer. Each modification includes a proposal of what the type 1 explanation should have been.
8. Select a modification. Choose an acceptable modification based on inconsistencies that were generated.
9. Repeat on underlying knowledge structures. At this point, the chosen modification indicates how a set of judgments and their type 1 explanations should be changed. For each judgment to be changed, the embedded knowledge structures that gave rise to the judgment need to be modified to produce the correct judgment and type 1 explanation.

To illustrate some these steps, consider this oversimplified example. The system chooses hypothesis h_1 with a rating of 8 out of 10 as its answer, and the correct answer rates h_2 with a 6. The question here is to decide how to modify the hypotheses' confidences - whether to increase them or decrease them. The qualitative model will produce an explanation showing how h_1 predicate missed the importance of some data item or possibly overrated itself by overweighting some supporting predicate, or how h_2 might have underrated itself by either underestimating the import of some piece of data or the impact of some predicate. The modification to be selected should result in the rating of h_2 higher than h_1.

[2]Much of this is admittedly vague, because our research is at an early stage.

The learning component will choose a modification from the following choices: including/excluding a predicate to be used in determining confidence in the hypothesis, lower or raise a hypothesis' confidence (or both), or increase/decrease the importance of a hypothesis' predicate. This modification implies that the decisions of underlying knowledge structures need to be changed; thus, these same steps will be applied to them also.

Other work has explored corrective learning using complex knowledge structures. SEEK [Politakis84] and SEEK2 [Ginsberg85], for example, perform corrective learning on structured collections of production rules. Both SEEK and SEEK2 look for statistical properties over a set of cases to discover and modify incorrect rules. This approach assumes that the correct conditions and conclusion for each rule have been identified, but that the logic combining these conditions or the confidence value produced by the rule might not be correct. By adopting an explanation-based approach instead, we intend to provide the capability to alter the conditions in a rule (or larger knowledge structure). Also, an explanation-based approach might lessen the the need for the kind of statistical analysis done by the SEEK programs.

Another example is ACES [Pazzani87], which uses device models for diagnostic reasoning and EBL. ACES uses a mathematical model of the device to confirm or reject fault hypotheses proposed by diagnostic heuristics. Rejected hypotheses cause the modification of diagnostic heuristics based on the reasons the model rejected it. Like ACES, our problem is a diagnostic one, but our system will differ in that our "diagnostic heuristics" will involve more complex problem solving structures and our device model will be qualitative and will be unable to categorically confirm or reject hypotheses.

Also, both SEEK and ACES assume that only one fault exists. As previously noted, this assumption does not hold in our domain. In fact, a CP patient usually has more than one malfunction.

In this paper, we have outlined our research plan to explore EBL techniques using explanations produced by complex problem solvers. Analysis of pathologic gait is complex because of multiple faults, the interactions between them, and the compensations for them. The analysis itself is the result of a complex problem solving process involving many different problem solving tasks. Several different types of explanations exist, and we plan to investigate how these different explanatory types impact an EBL approach to corrective learning.

References

Chandrasekaran, B. (1986). Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. *IEEE Expert, 1*(3), 23-30.

Chandrasekaran, B., Tanner, M. C., and Josephson, J. R. (1987). Explanation: The Role of Control Strategies and Deep Models. In Hendler, J. (Ed.), *Expert Systems: The User Interface*. Norwood, New Jersey: Ablex.

de Kleer, J., and Brown, J. S. (1984). A Qualitative Physics Based on Confluences. *Artificial Intelligence, 24*, 7-83.

Ginsberg, A., Politakis, P., and Weiss, S. M. (1985). SEEK2: A Generalized Approach to Automatic Knowledge Base Refinement. *Proc. Ninth International Joint Conference on Artificial Intelligence.* Los Angeles, Morgan Kaufmann.

Kuipers, B. J. (1986). Qualitative Simulation. *Artificial Intelligence, 29*, 289-338.

Pazzani, M. J. (1987). Failure-Driven Learning of Fault Diagnosis Heuristics. *IEEE Trans. Systems, Man, and Cybernetics, SMC-17*, 380-394.

Politakis, P., and Weiss, S. M. (1984). Using Empirical Analysis to Refine Expert System Knowledge Bases. *Artificial Intelligence, 22*, 23-84.

Simon, S. R. (1982). Kinesiology -- Its Measurement and Importance to Rehabilitation. In Nickel, V. L. (Ed.), *Orthopedic Rehabilitation.* New York: Churchill Livingstone.

WEINTRAUB-M@OSU-20

Dataproducts LZR 2665

```
OSU-20:WEINTRAUB-M  Job: LAIR-EBL.PS  Date: Fri Jan 29 15:20:47 1988

OSU-20:WEINTRAUB-M  Job: LAIR-EBL.PS  Date: Fri Jan 29 15:20:47 1988

OSU-20:WEINTRAUB-M  Job: LAIR-EBL.PS  Date: Fri Jan 29 15:20:47 1988

OSU-20:WEINTRAUB-M  Job: LAIR-EBL.PS  Date: Fri Jan 29 15:20:47 1988
```

A TASK ORIENTED APPROACH TO KNOWLEDGE-BASED SYSTEMS FOR PROCESS ENGINEERING DESIGN

Douglas R. Myers and James F. Davis*
Department of Chemical Engineering

David J. Herman
Department of Computer and Information Science

Laboratory for Artificial Intelligence Research
The Ohio State University
Columbus, Ohio 43210

*Author to whom correspondence should be sent

Table of Contents

ABSTRACT

Expert systems for process engineering design applications provide a means of capturing not only calculations but also the decision-making knowledge and efficient problem-solving methods of the design expert. Many important design applications in this domain involve design strategies and knowledge which are well-structured. Our task-oriented approach recognizes this structure in the design task and exploits it by describing the design task in terms of identifiable types of knowledge and a specific problem-solving strategy. DSPL (Design Specialists and Plans Language) is an expert system programming shell which allows knowledge characteristic of process engineering design problems to be explicitly represented according to the design task structure and offers an enhanced programming framework over first generation techniques emphasizing rule, frame and logic levels. STILL, an expert system for the design of sieve tray distillation columns, provides an application of the DSPL language and a demonstration of the methodology.

1. Introduction

Design plays a significant role in the process engineering industry. For a number of years, its importance has motivated much work in the development of computer-aided design (CAD) tools. These aids are used primarily to help the designer manage the many calculational, numerical, and database aspects of the design process. Little has been offered to automate the design process itself.

Although capable of greatly helping the designer, computer design tools have not offered a medium for effectively capturing the design procedure, nor uniting the host of different types of knowledge used in design. Experienced designers have certainly established efficient procedures for carrying out a design, and have acquired valuable decision-making knowledge used throughout the design. However, the current generation of design tools is generally used in a decision-support role rather than for decision-making. Considerable "how to" knowledge continues to be confined to the expertise of the human designer. These tools are hardly capable of being used as the foundation of a wholly automatic design system which produces a final product from initial specifications, unassisted by the engineer.

The inability of traditional programming techniques to effectively automate the non-calculational aspects of design has resulted in a shift toward the use of knowledge-based programming techniques to capture this layer of design expertise. A variety of rule-, frame-, and logic-based approaches have been employed with some success. These techniques hold the promise of capturing expert design knowledge which is difficult to represent by conventional means, with the benefit of making the knowledge widely accessible even though the expert may not be available. Additionally, there is the potential of freeing the expert from repetitious (in the view of the designer) yet complex design tasks. The benefits of knowledge-based techniques in design are exemplified by the success of systems such as XCON (Brug et al., 1986, McDermott, 1982), Pride (Mittal et al., 1986), and Micon (Birmingham and Siewiorek, 1984).

Much research has focused on characterizing the design process and understanding it as an intelligent activity. The majority of contributions are found in the artificial intelligence literature (e.g. (Balzer, 1981, Barstow, 1984, Mostow, 1985)). More recently, Stephanopoulos (Stephanopoulos et al., 1987a, Stephanopoulos et al., 1987b, Stephanopoulos, 1987a, Stephanopoulos, 1987b) has introduced the subject into the chemical engineering discipline. Drawing upon fundamental AI approaches, but focused specifically on process engineering applications, Stephanopoulos has discussed structure in design and demonstrated the use of object-oriented programming techniques. Our own work discusses the application of an alternative design methodology to capturing design knowledge in process engineering applications. The approach, described by Chandrasekaran (Chandrasekaran, 1986, Chandrasekaran, 1987), is a task-oriented view leading to an architecture defined in terms of a variety of knowledge types, organized and manipulated in ways specific to the design problem.

The scope of our work is limited to an important class of design problems in process engineering associated with well-defined knowledge and standard design methods. We refer to these problems as *routine design* problems (Brown and Chandrasekaran, 1986). For these well-defined problems, the task-oriented view provides a basis for categorizing pieces of design knowledge according to their role in problem solving.

Part of the motivation of the task-oriented approach is the need for a programming environment in which the terms of design problem solving can be properly articulated. Knowledge about the decomposition of design problems into subproblems, procedural information about problem solutions in the form of pre-enumerated plans, and constraints encoding design restrictions are all characteristic forms of knowledge in the design vocabulary, and as such should be directly describable in a programming environment. The task-oriented approach not only identifies these different types of design knowledge, but also helps determine how and where they should be used during the design process. By capturing the overall design strategy in the context of generic knowledge types, the task-oriented view offers a very specific framework for building design expert systems.

The Design Specialists and Plans Language (DSPL) is an expert system shell expressly suited for routine design problems (Brown, 1984). It was developed at the

Laboratory for Artificial Intelligence Research (LAIR) at the Ohio State University, and initially applied to the domain of mechanical design. In this paper, we demonstrate DSPL's usefulness in the process engineering domain as well.

DSPL provides an expert system building framework with a vocabulary which matches the terms of routine design problems. The language explicitly captures the types of knowledge and problem-solving strategies found in routine design. DSPL's ability to characterize the details of the design task makes it an appropriate language for building design expert systems for many applications in the process engineering domain.

We begin this paper with a discussion of the general issues defining well-structured design problems in the context of process engineering. The specific characteristics of "routine" design are identified, and the structure in such design problems is described. A presentation of the task-oriented view is followed by a discussion of the DSPL architecture and the types of knowledge and problem-solving strategies it makes explicit. Finally, our approach is illustrated with examples from STILL, an expert system for distillation column design.

2. A Perspective of the Design Process

There is little argument that design is a complex activity. The experienced engineer applies many different types of knowledge and problem solving strategies during the design process. However, many aspects of design are unknown, and many others are only poorly understood. When ill-defined concepts such as innovation and creativity are involved, the designer himself may have little initial conceptualization of how a design will eventually look or operate. It is certain that a complete understanding of the design process will not be available for some time.

However, many aspects of design can be agreed on. An experienced engineer often uses plans to help find a solution to a recognizable design problem. Often a plan is the result of a previous attempt to solve a similar problem in a similar situation. Plans which are successful are remembered and reused, while unsuccessful plans are either modified or discarded. The overall design normally proceeds from a more abstract level of description and representation to a more concrete, detailed level, and is often preceded with a sketch or rough design of the solution. Throughout the design, restrictions are checked to ensure that the design requirements are being met.

Numerical formulae are also an important part of a designer's knowledge. However unlike the design process itself, much work has been spent on investigating and developing knowledge about the mathematical relationships that hold within various applications. In general, the methodology for developing, representing and applying such quantitative knowledge is well-established. Our focus in this paper is on more symbolic forms of knowledge in the design process: knowledge which may decide when to apply a formula, rather than on the content of a formula itself.

Abstractly, design can be viewed as the selection of appropriate design attributes of a device or process and the subsequent specification of values for these attributes subject to various constraints on the design. The attributes may describe any type of design parameter, such as the physical dimensions of the device, or materials of construction. In more difficult forms of design, the final attributes of the device may not be known prior to the start of design. Indeed, even the functionality of the final device may not be initially well understood.

Constraints on the design also involve the attributes of the device. Size, cost, and operating constraints are all used at appropriate points in the design. The designer's task is to select values for each attribute, consistent with both the specifications of the problem and any constraints imposed by the domain itself.

Given a particular set of input requirements for a design problem, there are a potentially infinite number of possible values for the set of final design attributes. However, only a very few of these combinations, in fact, satisfy all the requirements of the particular application and constitute a solution to the design problem. The design problem can viewed as a search among the solution candidates for one of the suitable combinations.

Many methods could be employed to automatically search such a solution space. One simplistic method for search is the generate and test method. In this method the designer generates a potential solution by picking values for each attribute in the final design, and then tests the final design against each of the design constraints. This process continues until a solution is discovered which satisfies all of the design constraints. This method, however, is grossly inefficient and impractical, especially for complex designs requiring specification of many final design attributes. Experienced designers appear to use a more structured, organized strategy, relying on past experience to proceed rather directly and efficiently to a satisfactory solution to the design.

2.1. Routine Design

Within the spectrum of design, Brown and Chandrasekaran (Brown and Chandrasekaran, 1986, Chandrasekaran, 1986) have identified a range of design problems which they classify as "routine design". In part, the distinction between routine design and other types of design is the well-structured nature of the design procedure. Once the set of initial input requirements is given, the designer knows from past experience what design attributes must be specified and how the final design will look and operate. Well-defined design choices characterize each stage of the design process. Knowledge associated with each design choice is used to make appropriate decisions as the design progresses.

For this class of design problems, the overall structure of the particular device or process to be designed is essentially the same for each application. Each time, the same list of design attributes must be specified. However, the designer must produce a design that specifically matches the operational demands of the current application. For example, in designing a sieve plate in a distillation column, the conceptual structure of the plate is generally the same for each application, as well as is the number of attributes that must be specified. Only the actual values for the attributes change for each new application.

Even though a great deal of decision-making knowledge may be required to complete the design, the decision-making process at each stage is straightforward. This does not imply that routine design problems are trivial. The list of final design attributes that must be specified is known and finite, but the wide range of possible input requirements produces a very large number of potential final designs. The number of possible final designs even for apparently simple design tasks is sufficiently large to prohibit compiling a table of final designs from which a designer can look up the final design specifications. As a result, the design procedure, although often tedious, must be repeated for each new application.

2.2. Structure in Routine Design

In routine design problem solving, a variety of types of knowledge can be identified, each of which helps to efficiently solve portions of the design problem. The remainder of this section describes some of the forms of design knowledge in routine design which help to significantly simplify a design task.

Design Decomposition. The decomposition of the design problem into more manageable sub-problems is a key aspect of routine design. Through the experience of the designer, design sub-problems are established that can be solved relatively independently. Sub-problems typically correspond to the design of sub-assemblies or sub-systems of the device or process. In distillation column hardware design, for example, the design can be decomposed according to the major sub-sections of the column, such as the reboiler and the condenser. However, as a general comment, design sub-problems are not restricted to actual physical components of the device or process, nor do the sub-problems need to be completely independent.

The problem decomposition represents a "divide and conquer" strategy critical for efficient problem solving. The role of the problem decomposition can be illustrated by considering the interactions that might occur in a team of designers with a supervisor coordinating their design work. Suppose the supervisor is given a set of initial input requirements for the design of a process. The supervisor knows from experience that decomposing the design into the sub-problems "A", "B", and "C" facilitates the design process. Since each of the sub-problems can essentially be solved independently of each other. the supervisor assigns the responsibility of each sub-problem to each of three design engineers. With this arrangement. the supervisor acts as a coordinator to ensure that the design is executed in an appropriate order for a particular design situation. Furthermore, the supervisor handles any interdependencies between sub-problems by ensuring that the attributes assigned by each of the sub-problems meet the constraints of the other sub-problems.

For instance, if the designer for sub-problem "B" assigns a value for an attribute that does not satisfy a constraint for sub-problem "A", then the supervisor will have the designer for "B" perform some redesign. In this way. the supervisor makes certain that the parameters from the execution of sub-problems work together. Although a single designer will often perform the entire routine design alone rather than in an actual team of designers, the single designer still decomposes the design into sub-problems, working on these one at a time while making sure than the solutions for the sub-problems work together in a coherent overall design.

Problem decompositions typically exhibit several levels of abstraction. Near the top of the decomposition the nodes represent larger systems or assemblies, while the lower nodes represent clusters of components or particular components within an assembly. As a result, sub-problems near the top of the decomposition are more abstract in nature, while those nearer the bottom are more specific. The design process generally progresses from the top of the decomposition hierarchy to the bottom, similar to the behavior of the expert designer who focuses on broader issues early in design and avoids commitment to low level details until later.

Design Plans. Typically, one or more procedures for accomplishing the goal of each sub-problem are available. Different plans may address the same sub-problem, but under different assumptions or design conditions. One of the plans is selected for use during the design process. If the design is well-defined, the designer has knowledge about the context of the sub-problem to decide which design plan is appropriate. For example, a designer may have one plan for designing a sieve plate, another plan for designing a bubble cap plate, and so forth. The designer may choose the sieve plate plan because an input requirement requires a low pressure drop across the plate.

Each design plan consists of the computational and decision-making steps needed to specify values for all the attributes associated with the design goal of a particular sub-problem. These design steps can constitute a wide range of actions such as applying a design formula, numerically solving an equation, calling a simulation, or looking up a value in a table. Furthermore, decisions may be required in carrying out these actions. For instance, a decision would be required if several different formulae were available for carrying out a particular step under different design conditions, i.e., high pressure or low pressure.

Some of the design steps can be fundamental design decisions which assign a value to a design attribute. Design steps may involve numerical computations such as calculating plate active area or non-numeric decisions such as choosing the foaming tendency. A grouping of many design steps can represent a more complex coherent procedure such as a dynamic simulation or finite element analysis. Such complex procedures can either establish values for design attributes or can provide additional information for subsequent design steps.

Constraints. Constraints are tests performed by the designer involving one or more attributes pertaining to the design. In certain situations, the particular set of equations or procedures that the designer has developed will implicitly constrain the chosen attribute values. Frequently, however, the designer will have to explicitly check constraints during the course of the design.

Restrictions of one sort or another may apply throughout the entire design process. At the beginning of the design process, constraints can be used to check input requirements for suitability and can also be used to aid in design plan selection. Constraints may be used after the fact to ensure that all the final design attributes are satisfactory. From past experience, the designer knows which constraints are appropriate to aid in the design and when these constraints should be tested in order to properly constrain the design problem.

A designer often accumulates knowledge about what action to take when a violation of some restriction on the design is found. Typically, redesign of some portion of the partially completed design occurs involving changes to the values of several attributes of the design. Changes may be accomplished by changing the input to an equation used in design, or possibly using a completely different formula. Although there may be many ways to redesign a certain attribute. in routine design the designer knows how and where to accomplish the change. The designer knows which design attributes must be changed, what procedure should be used to invoke this change, and which attributes are affected as a result of this redesign. The design process proceeds only after redesign is completed in such a way that the constraint violation which triggered redesign is satisfied.

Unlike the first two types of knowledge described in this section, constraints don't so much shrink the design problem as simply help manage it. In non-routine portions of design problems, an engineer may not know how to test the suitability of a partial design, or even validate the characteristics of a completed one. In routine design, the means to verify the progress of a design are assumed to be available in some form as constraints.

The types of knowledge described in this section do not necessarily exist in all design problems: not all design problems fall into our category of routine design. However, we believe that a significant portion of design problems are in fact well-structured. Many problems exist in which experienced designers split a problem into smaller sub-problems, and then use design plans to solve the sub-problems. Our intention is to describe the knowledge structures inherent in this class of design problems, and illustrate how these structures impact the design process.

3. The Task-Oriented Approach

From the preceding discussion, it is apparent that there exist distinguishable types of knowledge used within routine design, and that there is a definite strategy for carrying out the design. Furthermore, the independence of the design strategy from any particular application indicates an underlying structure that generally describes routine design. Indeed, a definable organization and use of design knowledge has been found to exist in a variety of routine design applications. This organization is generic in that it seems to form the basis for organizing and carrying out routine design tasks.

In this paper, we the applicability of this task-oriented viewpoint to design problems in the processing plant domain. A number of important applications of interest to process engineers can be viewed as "routine design" problems. One broad class of applications which has been mentioned is distillation column design. Another important class is heat exchange equipment. Additional applications are associated with the design of other types of separation equipment and with certain types of reactors.

For any of these applications, specific types of knowledge and the application of the knowledge in the design process can be explicitly identified. The knowledge can be categorized in terms of four different types:

- Knowledge about the decomposition of the design problem into a hierarchy of manageable sub-problems. The knowledge is most effective when it results in sub-problems that have minimal design interactions.
- Procedural knowledge in the form of design plans which consist of individual design steps and constraint testing. Design plans also include appropriate redesign knowledge in the event constraints are not satisfied.
- Knowledge about the selection of appropriate design plans.
- Knowledge for adjusting the design in the event that a constraint is not satisfied.

The problem-solving strategy is one of coordinating the individual designs of the sub-problems to arrive at a consistent overall design. Communication between designers is established through a hierarchy of cooperating designers each responsible for an individual sub-problem. With respect to each design sub-problem, the context of the overall design drives the selection of an individual plan for accomplishing the design goal of the sub-problem. The plan is then executed. If a constraint fails, then available knowledge is used to try and adjust the design to satisfy the constraint. Once a design sub-problem has been completed, the answer is reported to the supervising designer.

This characterization of the design problem is representative of a more detailed analysis which has resulted in the development of a programming language specifically for the routine design problem (Brown, 1984, Brown, 1985). The following section describes some of the more important aspects of this language.

4. The DSPL Language

The Design Structures and Plans Language (DSPL) is a programming language tailored to the design task. It provides both a vocabulary for describing routine design and a built-in inferencing mechanism which uses the primitives of that vocabulary to advantage during the design process. The structures for organizing knowledge in DSPL not only facilitate the creation of routine design expert systems. but also define a general methodology for capturing the essential decision-making knowledge of the design process.

4.1. Programming Agents

DSPL represents design knowledge using a variety of programming constructs called *agents.* Each different construct is used to represent a different type of knowledge. A design hierarchy, for example, is used to describe the decomposition of a complicated design problem into simpler sub-problems. Plans are used to describe the courses of action the engineer pursues during design. Other constructs in DSPL capture the engineer's knowledge about how and when to choose appropriate plans. Constraints are used to represent knowledge about design specifications and relationships between various portions of the design. Each such construct provided by the language corresponds to some aspect of an engineer's expertise about solving design problems.

Specialists and the Specialist Hierarchy. The top-level programming agent in DSPL is the *specialist.* It is the unit of knowledge in DSPL which organizes almost every other kind of knowledge in the design system. Each specialist in a DSPL problem solver represents the knowledge for accomplishing the design of a particular sub-problem.

A problem solver in DSPL is built as a hierarchy of specialists which reflects the structure of the design problem. Typically, specialists higher up in the hierarchy deal with more general aspects of the process or device being designed. while specialists lower in the hierarchy deal with more specific design sub-tasks. The organization of the hierarchy mirrors the designer's expertise about dividing the design problem into smaller and simpler sub-problems.

In addition to representing the decomposition of the design problem, the hierarchical organization of the specialists also serves as a framework for coordinating the overall design. In DSPL, the top-most specialist acts as a supervisor, coordinating the design efforts of its sub-specialists. Any of its sub-specialists in turn can call upon their own sub-specialists to execute design sub-problems, and so forth. This specialist/sub-specialist organization is dependent on the sub-problem decomposition of the particular design application. When a specialist completes a portion of the design, the results are handed back to its parent specialist for further consideration. The design is complete when the top specialist in the hierarchy has received the results of all of its sub-specialists and successfully completed its own design decisions.

Each specialist is specifically responsible for its own design sub-problem and contains the local design knowledge necessary to accomplish that portion of the design. Several types of knowledge are represented. First, *design plans* in each specialist encode sequences of possible actions to successfully complete the specialist's task. Second, in the event a specialist has multiple design plans from which to choose, each design plan has an associated *sponsor* which contains knowledge about the appropriateness of its particular design plan. Third, in the event that more than one design plan is indeed available to a specialist, *plan selectors* within the specialist examine the run-time judgments of the sponsors and determine which of the design plans is most appropriate to the current problem. These three types of knowledge along with the structure of the specialist hierarchy are responsible for the focus of problem solving behavior during the course of the design process.

Plans and Plan Structure. A DSPL plan is a sequence of actions which a specialist uses to achieve its design goals. Each plan represents one method for completing the design sub-problem represented by the specialist.

The most basic design agent in a DSPL plan is the *step*. Each step is associated with selecting the value of one design attribute. For example, one step decides the derating factor for distillation column design, while another decides the tray spacing. The step contains whatever computations are necessary for selecting the value of the attribute. The value that the step selects may depend on the current state of the design, including any values previously stored in the design database by other steps in the system. Once the step decides on a value for its attribute, it stores the value in a design database.

A *task* in DSPL is an intermediate organizational structure between the step and the plan. Often, sequences of steps are related to each other in that taken together they perform a coherent aspect of the design plan. The task allows these related design steps to be executed together, and provides a convenient mechanism for clearly organizing steps within the design plan.

Constraints. *Constraints* may appear almost anywhere in the design hierarchy. They are generally used to check the relationship between the values of attributes in the design. These relationships may be numeric and involve a mathematical formula comparing the value of two attributes. Constraints often appear within a plan or task to verify some aspect of the design's progress. A constraint within a step may check the range of an intermediate value in the step, or check that the final value conforms to some input specification. A constraint may also appear as a pre-condition on a plan.

The failure of a constraint causes a redesign phase. During the redesign phase, previous design decisions are examined and possibly redone in an attempt to satisfy the failing constraint. Constraints contain suggestions for changing the design if the constraint test fails. These suggestions encode the domain knowledge needed to fruitfully direct the redesign process, rather than allowing unconstrained backtracking through all of the previous decisions made.

Redesigners. When a constraint is not satisfied, i.e. when the constraint test fails, the constraint provides suggestions concerning which attributes of the design can be profitably changed during redesign. The redesign actions are accomplished by programming agents called *redesigners*.

The redesign knowledge may be as simple as increasing or decreasing the value of a single design attribute or may involve more complex redesign such as using a different formula or procedure to change an attribute. Once these redesign changes are made, any attributes which depend on these redesigned attributes are automatically recomputed until the constraint can be tested again.

If there is no redesign knowledge associated with any-of the design attributes implicated by a failing constraint, or if the suggested redesign procedures are not successful, then the current design plan has failed in its goal of generating its portion of the design. This failure is reported to the specialist, and causes the specialist to discard the plan and attempt to select a new one from its collection of plans. The old plan's efforts are retracted, and the new plan is executed by the specialist. This process continues until either a plan succeeds or the specialist's supply of design plans is exhausted. If all plans have been tried unsuccessfully, the specialist's total failure is reported to its parent specialist.

4.2. Problem-Solving Strategy

The overall problem solving strategy in a DSPL system proceeds from the top specialist in the design hierarchy to the lowest. Beginning with the top specialist, each specialist selects a design plan appropriate to the requirements of the problem and the current state of the solution. The selected plan is executed by performing the design actions specified by the plan. These may include computing and assigning specific values to attributes of the device, running constraints to check the progress of the design, or invoking sub-specialists to complete another portion of the design.

For some types of design or constraint failures, the design process may be immediately terminated. In other situations, the engineer may have knowledge about how to repair the failure and continue with design. This kind of knowledge is encoded in DSPL as various *redesigners*.

The entire design process is complete when each specialist has executed a successful design plan to the satisfaction of the specialist's parent specialist. The top-most specialist makes the final decision if the design process is complete. At that point, a list of all final design specifications is made available to the user.

For more detailed discussion of the DSPL language, the reader is directed to the current literature on DSPL (Brown, 1984, Brown and Chandrasekaran, 1985).

4.3. Advantages of the Task-Oriented Approach

By providing generic programming structures specific for routine design, DSPL facilitates the building of expert systems. The different types of design knowledge and the usage of this knowledge in the overall context of the design is unambiguously captured by the various programming agents of DSPL. The template-like features within each of these agents simplifies insertion of appropriate design knowledge. All of the programming agents are organized to completely express the design strategy of the particular routine design problem. The resulting expert system exhibits predictable run-time behavior since all DSPL agents are used in an appropriate context of the design strategy.

The hierarchical architecture of DSPL allows explicit representation of the design knowledge and design strategy. As a result of this explicit representation, others can more readily use the expert system and understand the context of the design knowledge. The hierarchical problem-solving approach of DSPL also encourages creation of a modular system, to the extent that the particular design domain exhibits such modularity. This modularity together with the explicit representation enhances the maintainability of the expert system.

Current artificial intelligence (AI) programming approaches to building expert systems often involve rule-, frame-, or logic- based languages. These are useful as all-purpose programming tools, but by themselves they are often too general and unstructured, and make little commitment to a particular type of problem-solving. DSPL, on the other hand, represents a second generation AI language tailored specifically to the task of routine design. Limiting the language solely to the routine design task allows DSPL to contain programming structures specific to that task and greatly improves its leverage for routine design applications.

The lack of higher level, problem-specific constructs in many rule-, frame-, and logic- based approaches makes the building of expert systems analogous to programming in assembly language, where the programmer gains little support from the language in structuring a solution to a programming problem. At best, a disciplined programmer devises and enforces the use of useful structures on himself. At worst, the resulting program contains little structure at all.

5. STILL: An Expert System for Routine Distillation Column Design

Distillation column design is an activity in the domain of process engineering associated with considerable expertise and many different kinds of engineering knowledge. As a result, it has been identified as a potential expert system application (Rose, 1985). From a design task viewpoint, it is also a domain in which the characteristics of routine design often exist. The potential usefulness of expert system technology together with the characteristics of the problem have led to the development of STILL, a prototype expert system for the design of distillation columns.

Two facts support our claim that much of distillation column design is "routine". First, distillation column design is a well-established area of process engineering. There have been many years of developing, designing, testing, and operating distillation columns. Second, many examples and much discussion about column design have been documented. Indeed, much of the knowledge which is currently in STILL came initially from the open literature (Economopoulos, 1978, Henley and Seader, 1981, Van Winkle, 1967). Many methodological aspects of the design process were later verified in interviews with a practicing distillation column designer.

Our presentation of STILL in this paper is limited to simple, sieve tray columns. The examples serve to illustrate the task-oriented viewpoint and the use of DSPL for applications in process engineering. However, we are currently involved in extending the capabilities of STILL to different types of trays and more complicated column designs.

5.1. The Design Decomposition

As discussed in earlier sections, knowledge for decomposing the problem is a key type of knowledge which an expert designer brings into the design process. This decomposition knowledge is illustrated in STILL through consideration of the design process which begins with the hardware portion of the design. At this point in the design the complexity of the column, the type of tray, reboiler, condenser, materials of construction, performance requirements, etc., are specified. What is left in the design is the specification of the hardware parameters which describe the physical detail of the distillation column.

Figure 2 shows the specialist hierarchy of STILL which captures the decomposition of the design. The top specialist in the hierarchy is responsible for the complete design of the column and coordinates the activities of the sub-specialists. Each of the sub-specialists, Section, Reboiler, and Condenser, is responsible for the design of one major column component. The Section specialist, in turn, enlists the Plate Specialist to complete its portion of the design. The advantage of this representation is that the hierarchy explicitly shows how the overall design is organized into convenient sub-problems. Here the hierarchy expresses the strategy that the condenser, reboiler, and column section hardware designs can, to a large degree, be treated independently. Additionally, the hierarchy shows a connection between Section and Plate specialist expressing the fact that during the design of the column section, the hardware parameters of a tray need to be specified.

The interactions among the specialists in the hierarchy become more evident by viewing the design plans within each specialist. The design plans contain the procedural information for the specialist to accomplish its task. Figure 3 shows a plan from the Distillation Column specialist. A simplified DSPL syntax is used for illustration. The NAME clause gives the plan's name, "Simple Single Feed Design". The USED-BY clause shows that this plan belongs to the Distillation Column specialist.

The body of the plan appears in the TO DO clause. This clause lists the actions that are taken when the plan is used at run-time. It begins with a request to execute a task named "Validate Requirements". This task verifies that the input specifications for the column are valid for the methods and techniques used in the plan. The specifications which must be checked include such items as verifying that the components of the feed are hydrocarbons and checking that reasonable splits have been requested. The second action of the plan is a request to run a rigorous simulation. The column simulation establishes values for vapor and liquid flow rates, compositions, and temperatures in each of the trays as well as the condenser and reboiler duties. The DESIGN Section action results in a call to the Section specialist for a column design. Similarly, DESIGN Reboiler and DESIGN Condenser invoke the Condenser and Reboiler specialists. Since these are relatively independent sub-problems, the plan specifies that they may be done in parallel.

Figure 4 lists a design plan from the Section specialist. The run-time actions listed in the TO DO clause include requests to design a stripping section plate, the feed plate, and a rectifying section plate. Each plate design is accomplished by calling upon the Plate specialist. Here, since Section requires multiple plate designs, we see the advantage of expressing the plate design procedure explicitly as a specialist in the hierarchy.

This simple sequence of steps represents the actions that a process engineer takes when designing a distillation column. This piece of design knowledge is important not because this kind of knowledge is unusual (it certainly is not), but rather because it is made explicit in the problem solver and is represented at a level of detail which makes the knowledge and its use so obvious. Knowledge of this type represented in this fashion is typically easier to understand, debug and maintain than a cluster of rules which implement the knowledge uniformly without distinguishing between types and usages of knowledge.

5.2. A Detailed Design Plan

From Figure 2, it is clear that the design strategy progresses from general aspects of the design to detailed components. Accordingly, the plans which are associated with the Plate specialist contain the details of determining individual hardware parameters of the plate.

Figure 5 shows a graphic representation of a tray design plan within the Plate specialist. The plan contains the procedural information for completely designing one plate. The figure shows that the plan is decomposed into three tasks. Each is broken down into a number of individual steps. As suggested by the names, many of the steps are associated with the individual calculations of tray parameters and design variables, such as the downcomer area or the average width of flow path. When the plan is executed each of its design tasks is run in order. Within each task, each step is also run sequentially.

Figure 6 shows the DSPL code for the plan of Figure 5. The same language constructs are used as those illustrated for the Distillation Column and Section plans in Figures 3 and 4. The NAME, TYPE, USES and USED BY clauses in this plan are all used in a similar fashion. In this case the TO DO list consists of calls to the three tasks.

Similar language constructs are found in both the task and plan agents. Figure 7 shows the Final Tray Design task, which is the task most responsible for the design of the tray. This task consists of several design steps, a sub-task and several constraints.

5.3. Steps in a Design Plan

A distillation column designer also has knowledge for determining various attributes of the components of the distillation column. These fragments of design knowledge are represented in DSPL as *steps*. For example, in the tray design plan of Figure 5, the downcomer area step uses a mathematical formula, the the chord height step finds the root of an equation, and the tray spacing step uses a rule-of-thumb value.

To illustrate the template-like structure of a specific step agent, Figure 8 shows a DSPL step for determining the downcomer area of a tray. The name of the step is "Downcomer Area Designer". The USED-BY clause shows that this step is part of the Final Tray Design task. The REDESIGNER clause points to another DSPL agent, the Downcomer Area redesigner. The redesigner is used in the event that the value computed for the downcomer area is later found to be unacceptable.

The calculation for the downcomer area depends on a number of other attributes of the design, all of which are retrieved from the design database at the beginning of the step's execution. The step uses the values of these attributes in computing an intermediate value for the downcomer area, Adp. The step then chooses between Adp and a fraction of the active area, Aa, in determining the new value of the downcomer area. Finally, the new value is stored in the plate's database.

5.4. Constraint Testing and Redesign within a Design Plan

During the course of the tray design, the designer uses appropriately placed constraints to test the relationships among design attributes to verify that the design process is proceeding on track. For example, Figure 9 shows a constraint which is used to ensure that the plate's diameter is compatible with the plate spacing. The constraint fetches the current value of the plate's diameter, and decides which of four standard spacings is appropriate. The constraint checks that the selected value matches the existing value of the tray spacing in the design database. The relationship is not derived from an analytic model of the process, but rather is a rule-of-thumb based on the designer's experience. The positioning of this constraint in the Final Tray Design task (Figure 5) represents experiential knowledge in that the designer knows this is the appropriate place to perform such a constraint test.

Additional knowledge is needed for redesign in the event a constraint fails. The designer's experience dictates which previously determined attribute needs to be changed and how this redesign is to be accomplished. In the case of the Tray Spacing constraint (Figure 9), if the current tray spacing does not satisfy the constraint, the constraint suggests through the suggestions in its FAILURE-SUGGESTIONS clause that redesign should be accomplished by changing the value of the tray spacing. Control is then passed to the Tray Spacing redesigner (Figure 10). Redesign knowledge in this agent selects a new spacing using knowledge similar to the rule-of-thumb in the constraint. This knowledge is not used in the initial calculation of the spacing since the diameter has not yet been determined, and the dependencies preclude placing this computation before that of the tray spacing.

After the Tray Spacing redesigner has selected a new value for the tray spacing, any intermediate design steps between this redesigner and the tested constraint which depend on the tray spacing are automatically updated by the DSPL system. In this case, once the new tray spacing has been determined, the Downcomer Area, Total Tray Area, and Tray Diameter Design steps are executed again to update their values taking into account the new value for the tray spacing. At this point, the Tray Spacing constraint is again tested. If the constraint succeeds, the design proceeds. If the constraint fails on the second attempt, the Final Tray Design task fails, and further processing of the failure occurs at the next higher level in the Tray Design plan.

Constraint-testing and redesign in DSPL capture those methods used by human designers to proceed to a solution when purely algorithmic methods are awkward or simply not available. When a designer performs a routine design task and discovers that a design constraint has been violated, the designer knows from past experience exactly *which* design attributes must be changed (or "redesigned") and *how* to change their values. Furthermore, the designer knows that any design attributes depending on the newly-changed attributes must be recomputed. The DSPL architecture can take advantage of this kind of domain knowledge and relieve the designer from the details of representing all dependencies or otherwise requiring every possible combination of computations to be explored.

5.5. The Design Process In STILL

In STILL, the column input specifications are read in from an existing file, but they can also be collected interactively, either all at once before problem solving begins, or as needed by the system as the design process progresses. The specifications include the composition, pressure, temperature, and flow rate of the feed stream to the column, the light and heavy key components, and the desired light and heavy key splits. The following description illustrates the run-time behavior of the STILL system:

1. The design process begins when a design request is sent to the Distillation Column specialist. This activates the specialist and causes it to select and execute one of its design plans. The design plan of Figure 3 is currently the only design plan specified in the Distillation Column specialist. The strategy of this plan is fairly general, and suitably handles all of the design cases we are currently interested in. The plan's sponsor, the Default sponsor shown in Figure 11, is executed for the plan. Since the plan has not been previously used, it is designated as a "PERFECT" plan, i.e. the plan is perfectly suited to this design situation. The selector for the Distillation Column specialist is then run. Since the Simple

Column Plan is the only plan, and it is perfect for this situation, it is selected for execution.

2. As discussed in previous sections, each item in the plan is executed in turn. The input requirements are validated, and the rigorous simulation is run using the current specifications for input. The results of the simulation are made available to the rest of the design in a design database.

3. The next action in the Simple Column plan is a request to the Section specialist to perform its portion of the design. The execution of the Simple Column plan is suspended until this is complete, either with success or failure. At this point, the Section specialist controls the design process. The sponsors for each of its plans are requested to determine the suitability of their respective plans.

4. Figure 12 shows the sponsor for the Simple Section plan. As described in the previous section, the Simple Section plan depicts a very simple strategy for designing the section of a column. This strategy is only valid for certain process conditions, namely that the characteristics of the vapor and liquid molar flow rates are essentially constant in the process. The sponsor for the Simple Section plan checks for exactly these conditions, and decides on the suitability of the plan accordingly. If the molar flow rates are fairly constant, the plan is perfectly suited to the design situation. If the rates are highly variable, this strategy will not likely generate an acceptable design.

Assuming that the rates have a low variability, then the sponsor returns a suitability of "PERFECT" to the specialist. The specialist's plan selector takes this into consideration in its decision process, and selects this plan for execution by the specialist.

5. The execution of the Simple Section plan causes the Plate specialist to be invoked three distinct times, each with each invocation resulting in a plate being designed according to the data extracted from the rigorous simulation. The Simple Section plan is suspended while the Plate specialist performs its portion of the design, and regains control each time the Plate specialist finishes. The Final Section Design task is executed to adjust and integrate the individual tray designs into

a uniform column design, and finally control is returned to Simple Column plan in the Distillation Column specialist. At that point, the other portions of the column are completed as indicated by that plan.

Our existing STILL system runs on a Xerox 1109 workstation. The version of DSPL which we are using was implemented in LOOPS, an object-oriented programming system developed by Xerox on top of the Interlisp-D environment. Several pieces of the Generic Task Toolset including DSPL are also available in Intellicorp's KEE.

6. Discussion

The task-oriented approach to design differs from conventional equation-based techniques in two primary respects. First, a DSPL system attempts to capture the supplementary layer of problem solving knowledge that is beyond the calculational aspects addressed by conventional techniques. A system written in DSPL is an attempt to chart out the path of the expert's reasoning during design problem solving. On the other hand, conventional equation-based techniques are typically used to solve specific, closed problems. There is typically little flexibility in the application of the technique other than that introduced by the engineer applying them.

In STILL, equation-based approaches are used to determine design information such as tray temperatures, flow rates and concentrations, flooding condition and tray diameters. These are all well-tested calculational methods often used by distillation column designers. DSPL goes further than simply recording formulas to capturing knowledge about how and when the formulas are used during design.

Second, a DSPL system differs from conventional design programs in its ability to record the knowledge used during the expert's reasoning process. A DSPL system such as STILL is a kind of map of the pieces of knowledge used by the expert designer during routine design. The STILL specialist hierarchy, the specialist's design plans and the design steps all perspicuously document the distillation column design procedure. While many traditional design programs may be appropriate for certain aspects of a design problem such as distillation column design, and may even usefully solve certain subproblems, the resulting system would provide poor documentation of the design process itself. As discussed earlier, the constructs of DSPL facilitate understanding of the program and enhance the maintainability of the system.

The view that there exists a layer of problem-solving knowledge which determines the use of or interprets the results of calculations, then we see that DSPL is not a substitute for existing equation-based techniques. Rather, DSPL and traditional equation-based approaches are often complementary. For example, DSPL is not, in itself, an appropriate tool for optimization. If an optimization program of some sort is required during design, then DSPL is more properly used to coordinate the use of that method rather than as a tool for programming the method. We do not suggest the application of DSPL to problems where existing design techniques suffice.

It should also be pointed out that an expert's design knowledge does generally lead to a "best" design. DSPL is an attempt to capture a designer's strategy in a more tractable form. This tractability is traded off against the "certainty" of a closed form which would take an inordinate amount of time to compute. In this case, though, "best" is determined through the experience of the designer and not in a mathematical sense.

Issues surrounding appropriate mathematical definitions of design problems are peripheral to the task-oriented approach. In the context of routine design, if the problem is well-structured and the equation-based techniques are correctly applied so that the expert designer can arrive at a solution, then we conclude that the design problem is appropriately defined. If this is not the case then the problem may not be routine design, or the "expert" is not expert, i.e. the equation-based techniques are not being used properly.

7. Conclusions

An essential layer of problem-solving knowledge in process engineering design is comprised of efficient strategies and decision-making information. This layer of design expertise exists over and above the calculational aspects of design. Expert systems are considered the most viable approach to capturing design information contained in this layer. They not only provide a means of capturing qualitative design knowledge, but also offer a medium for exploiting the efficient methodologies used by design experts.

In this paper we present a framework for building design expert systems which effectively captures both design knowledge and problem-solving strategies which are found in process engineering domain applications. The approach, referred to as the "task-oriented" approach, is based on the identification of the various types of knowledge used and the definite structure of the methodology. Our goal is the development of design expert systems which can carry through to the completion of a design. It is shown that the approach is applicable to well-structured design problems. In the process engineering domain, this class of design problems represents an important set of potential applications.

Since it identifies the knowledge types and problem-solving structures underling the routine design task, the task-oriented approach provides an applications-independent view of design. This framework is made explicit in DSPL (Design Specialists and Plans Language), a programming language which offers specific constructs for representing each of the identifiable types of knowledge found in the design task and inferencing strategies for taking advantage of that knowledge. Because of the applicability of the task-oriented view to certain process engineering design problems, DSPL provides a programming environment which greatly facilitates the development of design expert systems in this domain.

Additionally, this task-oriented framework provides the medium for articulating the design methodology at an appropriate level of understanding. This helps during the development of the expert system for knowledge acquisition and also aids in the maintenance and usability of the system.

8. Acknowledgments

At the Laboratory for Artificial Intelligence Research (LAIR), grants 82-0255 and 87-0090 from the Air Force Office of Scientific Research, as well as support from the Defense Advanced Research Projects Agency, RADC Contract F 30602-85-C-0010 have played an important role in the development of these ideas. We would also like to acknowledge valuable equipment support from the Xerox University Grant program and financial support from the NSF and Du Pont Fellowship programs.

9. Bibliography

Balzer. R. (January 1981). Transformation Implementation: An Example. *IEEE Trans. Software Engineering, SE-7*(1). 3-14.

Barstow, D. (1984). A Perspective on Automatic Programming. *AI Magazine*, *5*(1), 5-27.

Birmingham, W. and D. Siewiorek. (June 1984). Micon: A Knowledge-Based Single-Board Computer Designer. *Proc. ACM/IEEE 21st Design Automation Conference*. .

Brown, D. C. (1984). *Expert Systems for Design Problem-Solving Using Design Refinement with Plan Selection and Redesign.* Doctoral dissertation. Ohio State University.

Brown, D. C. (August 1985). Capturing Mechanical Design Knowledge. *Proceedings of ASME CIME*. .

Brown, D.C. and B. Chandrasekaran. (April 1985). Plan Selection in Design Problem-Solving. *Proceedings of the AISB85 Conference*. The Society for AI and the Simulation of Behavior (AISB), Warwick, England.

Brown, D. C. and B. Chandrasekaran. (July 1986). Knowledge and Control for a Mechanical Design Expert System. *IEEE Computer, 19*(7), 92-100.

Brug, A., J. Bachant and J. McDermott. (Fall 1986). The Taming of R1. *IEEE Expert, 1*. 33-39.

Chandrasekaran, B. (Fall 1986). Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. *IEEE Expert*. *1*. 3-30.

Chandrasekaran, B. (August 1987). Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks. *Proceedings of IJCAI-87*. IJCAI.

Economopoulos, A. P. (December 1978). Computer Design of Sieve Trays and Tray Columns. *Chemical Engineering, 85*(27), 109-120.

Henley. E. J. and J. D. Seader. (1981). *Equilibrium-Stage Separation Operations in Chemical Engineering*. New York: John Wiley & Sons, Inc.

McDermott, J. (September 1982). R1 - A Rule-Based Configurer of Computer Systems. *Artificial Intelligence, 19*(1). 39-88.

Mittal, S., C. L. Dym and M. Morjaria. (1986). PRIDE: An Expert System for the Design of Paper Handling Systems. IEEE Computer Special Issue on Expert Systems for Engineering Applications.

Mostow, J. (Spring 1985). Towards Better Models of the Design Process. *AI Magazine*, *6*(1), 44-57.

Rose, L. M. (1985). *Computer-Aided Chemical Engineering*. Vol. 1: *Distillation Design in Practice*. New York: Elsevier Science Publishing Company, Inc.

Stephanopoulos, G. (April 1987). Process Control with a Computer. *CHEMTECH*, *17*(4), 251-256.

Stephanopoulos, G. (September 1987). The Future of Expert Systems in Chemical Engineering. *Chemical Engineering Progress*, *83*(9), 44-51.

Stephanopoulos, G. and T. Kriticos. (March 9-10 1987). A Knowledge-Based Framework for Process Design and Control, Part I: An Artificial Intelligence Perspective in Design. *Workshop on Artificial Intelligence in Process Engineering*. Columbia University, New York: NSF-AAAI.

Stephanopoulos, G. et al. (1987). DESIGN-KIT: An Object-Oriented Environment for Process Engineering. *Comput. chem. Engng.*, *11*(6), 655-674.

Van Winkle, M. (1967). *Distillation*. New York: McGraw-Hill, Inc.

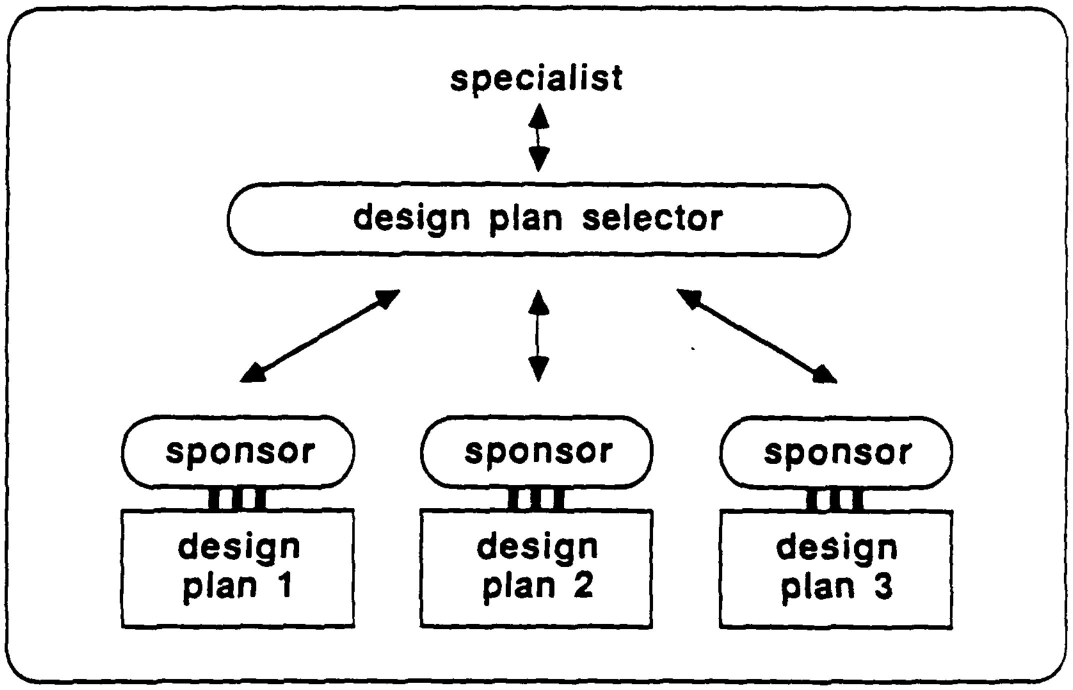


Figure 1: The structure of a specialist.

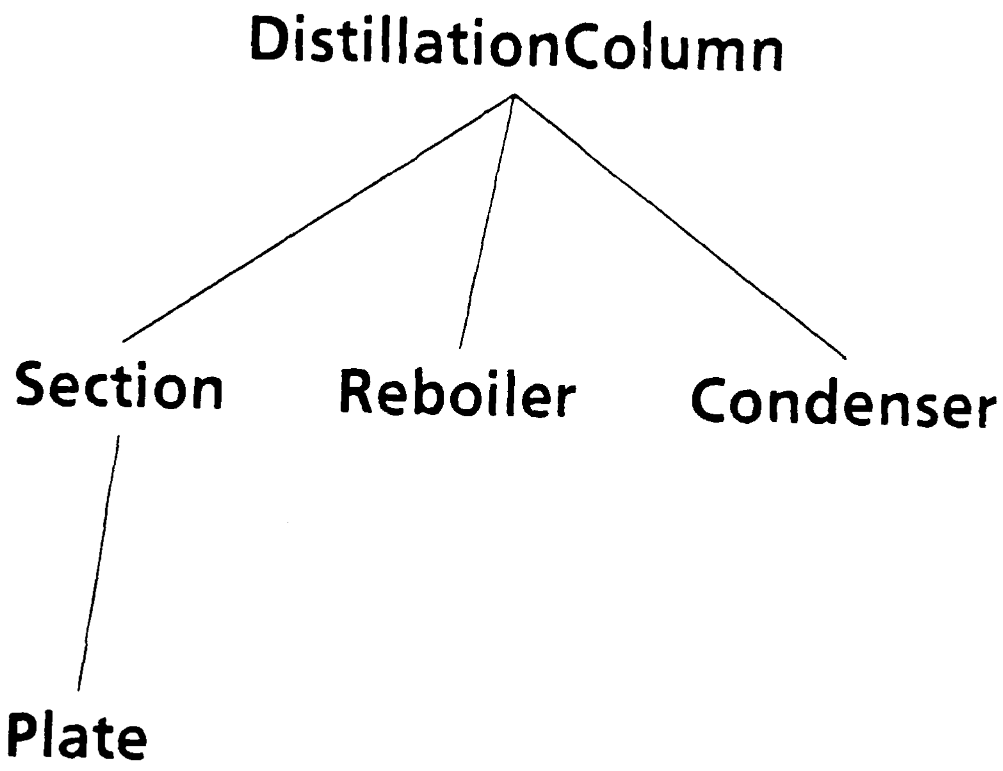


Figure 2: The STILL specialist hierarchy.

```
PLAN
   NAME    Simple Single Feed Design
   TYPE    Design
   USES    Section, Condenser, Reboiler SPECIALISTS
   USED BY Distillation Column SPECIALIST
   SPONSOR Default SPONSOR
   TO DO
      Validate Requirements
      Invoke Rigorous Simulation
      DESIGN Section
      PARALLEL   DESIGN Reboiler
                 DESIGN Condenser
```

Figure 3: A plan for column design.

```
PLAN
   NAME    Simple Section Design
   TYPE    Design
   USES    Plate SPECIALIST
   USED BY Section SPECIALIST
   SPONSOR Simple Section SPONSOR
   TO DO
      DESIGN Stripping Plate
      DESIGN Enriching Plate
      DESIGN Feed Plate
      Final Section Design
```

Figure 4: A section design plan.

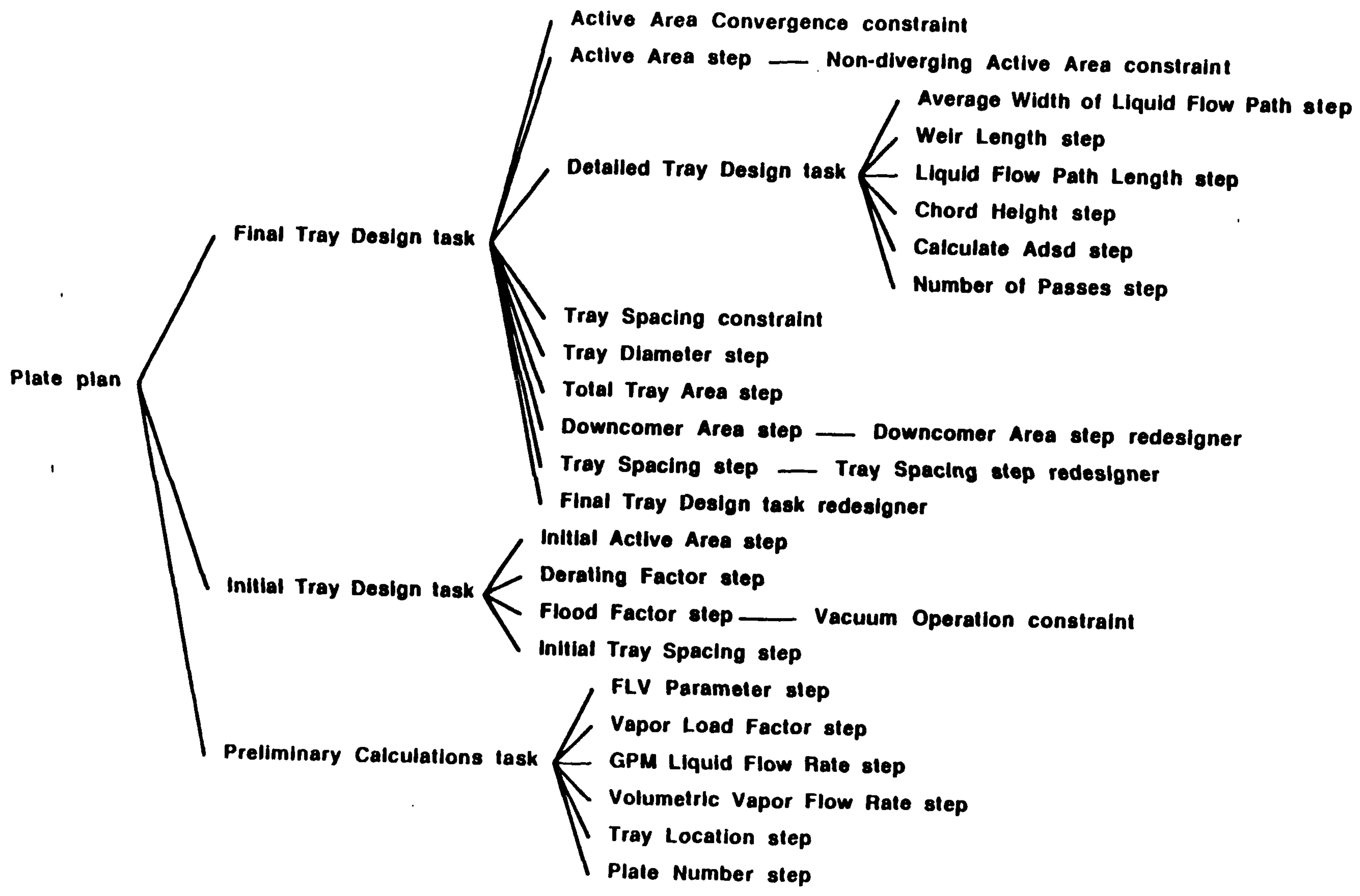


Plate plan
Final Tray Design task
Active Area Convergence constraint
Active Area step
Non-diverging Active Area constraint
Detailed Tray Design task
Average Width of Liquid Flow Path step
Weir Length step
Liquid Flow Path Length step
Chord Height step
Calculate Adsd step
Number of Passes step
Tray Spacing constraint
Tray Diameter step
Total Tray Area step
Downcomer Area step
Downcomer Area step redesigner
Tray Spacing step
Tray Spacing step redesigner
Final Tray Design task redesigner
Initial Tray Design task
Initial Active Area step
Derating Factor step
Flood Factor step
Vacuum Operation constraint
Initial Tray Spacing step
Preliminary Calculations task
FLV Parameter step
Vapor Load Factor step
GPM Liquid Flow Rate step
Volumetric Vapor Flow Rate step
Tray Location step
Plate Number step

Figure 5: A graph of a detailed design plan.

```
PLAN
   NAME    Tray Design
   TYPE    Design
   USES    No SPECIALISTS
   USED BY Plate SPECIALIST
   SPONSOR Default SPONSOR
   TO DO
      Preliminary Calculations
      Initial Tray Design
      Final Tray Design
```

Figure 6: The tray design plan.

```
TASK
   NAME    Final Tray Design
   USED BY Tray Design PLAN
   TO DO
      Tray Spacing
      Downcomer Area
      Total Tray Area
      Tray Diameter
      TEST-CONSTRAINT Tray Spacing and Diameter Compatible?
      SUB-TASK Detailed Tray Design
      Active Area
      TEST-CONSTRAINT Active Area Converged?
```

Figure 7: A task.

```
STEP
   NAME    Downcomer Area
   COMMENT Dependent on flow rates, densities, Active
           Area,   etc.
   USED BY Final Tray Design TASK
   ATTRIBUTE-NAME  Downcomer Area
   REDESIGNER      Downcomer Area REDESIGNER
   TO DO
   KNOWNS  FETCH Plate Active Area
           FETCH Plate Liquid Flow Rate
           FETCH Plate Flood Factor
           FETCH Plate Derating Factor
           FETCH Plate Tray Spacing
           FETCH Plate Liquid Density
           FETCH Plate Vapor Density

   DECISIONS
      Vd  IS SMALLEST OF  250.0 * Sf  AND
              7.5 * (Sf * SQRT (Ts * (Pl - Pv)))  AND
              41.0 * (Sf * SQRT (Pl - Pv))
      Adp IS LGPM / (Vd * Ff)
      Downcomer Area IS LARGER  OF  Adp  AND
              SMALLER OF  Aa * 0.11  AND  Adp * 2.0
      STORE Plate Downcomer Area
```

Figure 8: A step.

```
CONSTRAINT
   NAME    Tray Spacing and Diameter Compatible?
   COMMENT Checks to see if the tray spacing is appropriate.
   USED BY Final Tray Design TASK
   FAILURE-MESSAGE The current tray spacing is inappropriate
                          for the tray diameter
   FAILURE-SUGGESTIONS     CHANGE Tray Spacing
   TO DO
   KNOWNS  FETCH Plate Tray Spacing
           FETCH Plate Tray Diameter
           Best Spacing IS DEPENDENT-ON Tray Diameter:
                IF < 3.0 THEN 12.0
                IF < 5.0 THEN 18.0
                IF < 6.0 THEN 24.0
                IF < 8.0 THEN 30.0
                OTHERWISE FAIL
   TEST Current Spacing = Best Spacing?
```

Figure 9: A constraint.

```
STEP-REDESIGNER
   NAME    Tray Spacing
   COMMENT Changes the tray spacing based on the tray diameter
   USED BY Tray Spacing STEP
   VALUE TO CHANGE
           Plate Tray Spacing
   CHANGE
   KNOWNS  FETCH Plate Tray Diameter

   DECISIONS
           Tray Spacing IS DEPENDENT-ON Tray Diameter:
                IF < 3.0 THEN 12.0
                IF < 5.0 THEN 18.0
                IF < 6.0 THEN 24.0
                IF < 8.0 THEN 30.0
                OTHERWISE FAIL
           STORE Plate Tray Spacing
```

Figure 10: A step redesigner.

```
SPONSOR
   NAME    Default
   USED BY Default SELECTOR
   TO DO
      IF PLAN ALREADY TRIED THEN RULE-OUT
      ELSE SUITABLE
```

Figure 11: A Default plan sponsor

```
SPONSOR
   NAME    Simple Section
   USED BY Default SELECTOR
   PLAN    Simple Section PLAN
   TO DO
   KNOWNS  FETCH Molar flow rate data
           Variability of molar flow rate data

   DECISIONS
      SUITABILITY IS DEPENDENT-ON Variability
           IF LOW THEN PERFECT
           IF MODERATE THEN DONT-KNOW
           OTHERWISE RULE-OUT
```

Figure 12: The Simple Section plan sponsor.

An Information Processing Model of Japanese Foreign and Energy Policy Decision Making: JESSE

Donald A. Sylvan

Department of Political Science
The Ohio State University

Ashok Goel and B. Chandrasekaran

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

February 1988.

This paper will be presented at the Annual Meetings of the International Studies Association, St. Louis, Missouri, March 30-April 2, 1988. An earlier version of this paper was presented to the Seminar on Foreign and Defense Policy Decision Making, The Mershon Center, The Ohio State University, January 27-28, 1988.

An Information Processing Model of Japanese Foreign and Energy Policy Decision Making: JESSE

Donald A. Sylvan, Ashok Goel, and B. Chandrasekaran

ABSTRACT

This article contrasts information-processing approaches to decision making with other approaches to understanding foreign policy decision making. After examining the type of political domains for which information processing approaches are likely to be helpful, the article proposes an information processing based theory of Japanese foreign policy making. That theory is embodied in an experimental system called JESSE, that models decision making by the Japanese political and economic elite in the domain of her energy supply security. The system is initiated by supplying information about an energy-related event. It recognizes the threat posed by the event to Japanese energy supply security, and delivers a set of plans appropriate for the situation. In deciding on a set of plans, the system takes into account the state of Japanese foreign relations which impose constraints on the choice of policy options. JESSE contains multiple modules that perform the generic information processing task of Classification, and a module that performs the generic task of Plan Selection and Refinement. JESSE is tested in a number of ways, including the case of the Iranian revolution. It is found to be a quite plausible model.

1. Introduction

In recent decades, political scientists have put forth a number of frameworks, pre-theories, and even theories about the making of foreign policy decisions. Institutional approaches, bureaucratic politics approaches, and a multitude of others are by now familiar to us all. This article sets forth an information processing based approach as a supplement to those other conceptualizations. We do not claim that an information processing approach is appropriate for understanding all types of political or foreign policy decision making. Rather, we argue that domains that have certain political characteristics are more likely to be productively treated by an information processing approach. Borrowing from [Sylvan, 1987], the scope conditions for applicability of an information processing approach are the identity of the political unit being modelled, the depth of the modeller's understanding of the political unit's problem solving behavior, the question of whether the political unit can be reasonably characterized as exhibiting an identifiable general mode of problem-solving, and whether there is sufficient information available to provide for a validity test. As will be discussed later in this article, we see the domain of Japanese supply security decision making as meeting these criteria. We would not find, for instance, certain aspects of U.S. foreign policy decision making to meet these criteria. Such features of the Japanese supply security case as an identifiable general mode of problem solving make that case an ideal one to analyze through an information processing approach.

The core of our argument is that some domains of political decision making - including the one we address here - seem to behave as an information processing model would predict. Some of the fundamental conflictual elements of politics have been resolved either prior or exogeneously to the onset of the decision domains in question. Goals are often quite clear in this subset of decision environments, oftentimes because they include maintenance of what are perceived to be essential functions of the polity. As a result, the process of decision making unfolds as information processing theory would expect. Our political science judgment is that Japanese supply security is such a domain. We, therefore, propose, explicate, and test an information processing model of Japanese foreign and energy policy decision making here.

1.1. Comparison to other Approaches

Scholars and observant lay people alike have been impressed with Japanese economic performance in recent decades, especially in the context of the political, military, annd natural resource obstacles that Japan faces. In the political science community, students of both foreign policy decision making and of international political economy are potential sources of explanation of this success. In the area of international political economy, the writings of such neo-Marxian scholars as [Wallerstein, 1984] are one place to look. Such writings would lead one to seek an explanation of the contrast between the success of one capatilist nation-state - Japan - and the more difficult ecconomic situation of other capatilist nation-states such as the United States by focusing almost exclusively on the state of hegemonic status of these two nations. To us, this is an unsatisfactory and somewhat post-hoc explanation. Liberal economists[1] on the other hand, offer us the basis for a contrasting, but equally incomplete explanation. Their image of relatively unconstrained nation-states acting on laws of supply and demand omits a great many factors.

Realists and neo-realists in international relations from [Carr, 1946] through [Morgenthau, 1966] to [Krasner, 1978] have difficulty explaining Japanese influence and success given the lack of large military expenditures.

[1] See for instance the modelling of economic sectors in [Bremer 1987]

A fourth alternative is more at home in the study of foreign policy decision making than in the study of international political economy. It is based upon a conception of Japan, and other nation-states, as making decisions under constraint. This conception views Japan as a nation-state that has very real resource constraints, and succeeds more than do other nation-states by approaching the constraints in a novel manner. Such a view serves as the starting point for our research. We argue that political science can enrich its understanding of decision making by learning from the study of artificial intelligence (AI) and cognitive science. Many studies of comparative foreign policy have concentrated on either the identity and nature of institutions and structures of government or on the actions of individual bureaucrats (e.g., [Allison, 1971]). We argue, in contrast, that the reasoning of political and economic elites who have had similar political socialization can be captured as a "group cognition." This group cognition subsumes the working of institutions of government, and accounts for a great deal of political decision making.

1.2. Overview

Almost all of artificial intelligence research on cognitive modelling has been concerned, implicitly in most cases, with *individual* human cognition, for instance human problem solving and planning. However, *organized collectives* of humans also solve problems, and synthesize plans. Indeed, organized collectives, such as national political elites, perform many of the functions and display many of the behaviors that, typically, we associate with individual humans. Political and economic elites of nation-states, for instance, not only engage in problem solving and planning, but also retrieve information from memory, learn from experience, and explain their behavior among other, similar information processing activities. Might we then productively ascribe a "mind" to at least some organized collectives of humans? Might we consider national political elites to be "intelligent"?

We believe that a study of political cognition in the AI paradigm may yield important clues to a better understanding of social intelligence, and thus of intelligence in general. While there are significant differences with in the AI research community on the constitution of the AI paradigm, there is also substantial agreement on the importance of the role of knowledge in cognition. Indeed, information processing theories of representation, organization, and use of knowledge have been long playing a central role in understanding individual cognition. It seems obvious to us that modeling political cognition, for instance decision making by national political elites, also should provide a rich arena for experimentation with theories of knowledge. Further, theories of knowledge representation and organization are likely to provide *languages* for expressing theories of some political phenomena, for instance international relations. The development of such representation languages may be expected to provide precision to some political theories, and impose a discipline on them. Moreover, it may allow for testing the theories to some degree by computational experimentation with them.

Our approach to decision making is based on a theory of *generic information processing tasks* for understanding knowledge-using reasoning, and construction of knowledge-based systems [Chandrasekaran, 1986; Chandrasekaran, 1987]. The theory proposes that complex information processing tasks, such as decision making, often are performed by decomposition into a small set of generic tasks. A generic task is a "natural kind" of information processing task, corresponding to which is a primitive type of reasoning that provides a basic building block of intelligence. Classification, and Plan Selection and Refinement are two examples of generic tasks. A generic task, such as Classification, is characterized by the information processing function of the task, the representation and organization of knowledge needed for performing the function, and the control strategy that accomplishes the function. The knowledge and control structures used in the performance of each generic task are such that its functionality can be achieved computationally efficiently.

In this paper, we report on an experimental knowledge-based system called JESSE, for Japanese Energy Supply Security Expert[2], that models some aspects of Japanese energy policy decision making [Goel and Chandrasekaran, 1987; Goel*et al.*, 1987]. The system is initiated by supplying information about an energy-related event, such as the Iranian revolution of 1979. It recognizes the threat posed by the event to Japanese energy supply security, and delivers a set of plans appropriate for the situation. In deciding on a set of plans, the system takes into account the state of Japanese foreign relations which impose constraints on the choice of policy options. Thus, JESSE performs the complex information processing task of constrained decision making which involves the tasks of threat recognition, constraint formulation, and reactive planning.

The rest of the paper is organized as follows: In the next section, we specify the epistemic basis of our work. We present an analysis, and a model of Japanese energy policy decision making in sections *3* and *4*, respectively. In section *5*, we discuss some of the assumptions, limitations, and implications of our research. We conclude the paper in section *6*. However, before we proceed further we need to caution the reader that since our research lies at the intersection of Political Science and AI we have a problem with the proper usage of terminology. For those terms for which a conflict of academic traditions arises, we will use terms in accordance with their common usage in the political science community. Two examples of such conflicts are: What we call "group cognition" is sometimes known as "collective cognition" in AI. Similarly, what we call "computational models" below are sometimes referred to as "AI models" in the AI literature.

2. Information Processing Models of Political Cognition

There is a small, but growing body of literature on AI models of political cognition. We will not provide here a comprehensive survey of these models. Instead, we confine our attention to only those issues that help specify the epistemic basis for our work on Japanese energy policy decision making.

2.1. Models of Political Decision Making

Since the early 1970's a number of *computer simulation* models have been developed [Meadows *et al.*, 1982]. These models are strictly neither computational models, nor models of political cognition *per se*; we mention them here only to contrast our work with them. There are two main characteristics of computer simulation models. Firstly, they are based on the classical rational decision making theories, in which the causal relationships between the arguments is independent of an understanding of the agents. Secondly, their domains, typically, are global in scope. Since the late 70's several computational (or AI) models of political decision making have been developed. In contrast to the computer simulation models, the actions of the agents in the AI models are based on intentional inferencing, or goal-directed behavior, rather than on rational decision making. We believe that the case for models based on intentional inferencing over models based on rational decision making as a better description of the political cognitive process has been well established [Simon, 1985; Sylvan, Bobrow, and Ripley, 1987].

Computational models of political cognition may be classified into *computational linguistic* models, and *information processing* models. The computational linguistic models are based on text interpretation and discourse analysis which map a political discourse into a set of arguments [Carbonell, 1981; Mallery, 1987], while the information processing models attempt to understand the

[2]While "Expert" fits well as part of our acronym, the rationale for our research is not building a usable expert system for policy making, though that may be a useful by-product. Instead, we seek to construct, test, and refine an information processing theory of foreign policy decision making.

political decision making as a problem solving and planning activity [Thorson, 1984; Mefford, 1987; Sylvan, 1987; Majeski and Sylvan, 1987; Job, 1987]. We believe that it is important to first have a good theory of the *mechanism* by which political decisions are made, that will then yield the proper set of arguments into which a political discourse may be mapped by linguistic analysis. Nevertheless, our approach is compatible with, and complementary to, the approaches based on text interpretation and discourse analysis.

2.2. Levels of Aggregation of Political Units

One of the dimensions in which the various formulations of policy decision making differ from one another involves the level of aggregation of the political unit being modeled. Typically, the choice is between notable single individuals, or some particular organizations, or other elites. We are interested in understanding the process by which an ensemble of actors, collectively labeled a national government in the name of a nation-state, arrives at decisions. Our work seeks to provide an architecture ("functional" in AI terminology) for the decision making political actor that spans the particular individuals and institutions of implementation. Thus, we have chosen to focus on a national political and economic elite as the level of aggregation.

Policy decision making by such a national political and economic elite is an instance of what we shall call "group cognition." We shall contrast our view of group cognition with two alternative views that we will discuss together under the rubric of "collective cognition." In our view, a group (in this case Japanese foreign policy decision making elite) is seen as having a cognition, and we are attempting to represent that cognition.

Collective cognition, by contrast, can take on two forms. In one form, typified by [Lau and Sears, 1986], individual political actors, such as voters, are examined for their individual cognitions. The results of that examination can then be aggregated into a profile of a larger entity, such as a nation, to create what can be termed, for example, a collective American cognition. Another alternative can be found in work by [Majeski and Sylvan, 1987][3]. In this way of looking at collective cognition, a single decision maker is studied and his or her cognition modelled. An example from Majeski and Sylvan's work would be modelling Walt Rostow as part of understanding U.S. decision making *vis a vis* Vietnam. To arive at a collective cognition, one would then have to model each key decision maker and then posit some manner of combining those models to produce a model of a decision. One might assign weights to each decision maker (e.g., [Shapiro and Bonham, 1982] on cognitive mapping. A second possibility would be to posit some combinatory rules based upon a sophisticated theory of group dynamics from social psychology. Since we have not found a social psychological theory that we wish to embrace for these purposes, we find the notion of group cognition more helpful. Our understanding of Japanese foreign policy decision making in particular leads us to believe that political socialization makes the assumption of a single group cognition a reasonable approximation.

2.3. Levels of Information Processing Abstractions

Another dimension in which the various computational models of political decision making differ from one another concerns the level of abstraction at which the actions of the decision maker are described. Typically, predicate logic or production rules have been the preferred levels. Threat recognition by nation-states, for instance, has been modeled at the level of logic [Gaucas and Brown,

[3]Note, however, that more recent work by these scholars more closely approximates what we here are calling group cognition

1987], and rules [Lenat *et al.*, 1983]. We believe that these levels of abstraction are too low for properly understanding policy decision making. At these levels, the semantics of information processing often is lost, and the decision maker is viewed mainly as a syntactic manipulator of logical predicates or production rules. Instead, as we alluded earlier, there exist generic information processing tasks, with corresponding strategies, which provide a high-level language for characterizing decision making. Indeed, from this level of abstraction, the logic and the rule level mechanisms may be thought of as ways of *implementing* the higher-level strategies.

A related issue arises with the choice between case-based reasoning and "compiled" reasoning. We have used the compiled form of reasoning in our work, where both the threats to energy supply security into which an energy-related event is mapped, and the plans that are indexed by these threats, are available in a compiled form. However, the threat types and plans that we use are themselves higher level abstractions of a large number of individual cases *i.e.* they are compiled prototypes of individual cases. The difference is that instead of searching through a large number of individual cases, only the space of higher level abstractions needs to be searched. Our approach emphasizes cognitive structure over cognitive content. This means that we first examine the knowledge organizations and control regimes that are used in political decision making before examining the specific knowledge that is used. However, our approach is compatible with, and complementary to, case-based reasoning.

3. An Analysis of Japanese Energy Policy Decision Making

Japan is a country that is poor in natural resources such as energy. She is critically dependent on energy exporting countries for her energy needs. In recent years, the world energy situation has been volatile, for instance the massive increase in the cost of energy following the Iranian revolution in 1979. How does Japan reason about an energy-related event such as the Iranian revolution?

We posit that Japan has prepared in advance policy options for anticipated threats to her energy supply security [Bobrow *et al.*, 1986; Sylvan *et al.*, 1987]. Some of the stored policy options are unilateral (*e.g.* buy energy shares in the stock market), some are bilateral (*e.g.* purchase energy from reliable energy exporting countries), while others are multilateral (*e.g.* support multilateral energy consumer cartels). How does Japan select appropriate policy options in response to an energy-related event?

Japanese energy policy decision making takes place in the context of her foreign relations in general. At the time of the Iranian revolution, for instance, Japan relations with some far east Asian countries were strained. What is the role of Japanese foreign relations generally in her energy policy decision making?

We would like to argue that an implicit goal of Japan is to maintain a low cost supply of imported energy commensurate to her energy needs. Some of the various energy-related events that occur in the world may threaten this goal. In principle, when an energy-related event occurs, Japan may use the event as an index to the policy options that she has prepared in advance. However, since the number of energy-related events that might occur in the world is very large, a direct mapping of the events onto the policy options would be, in general, computationally very expensive. This task may be performed more efficiently by decomposing it into two tasks as follows: Firstly, energy-related events may be *classified* onto a small number of stored categories. Each category is an equivalence class of some subset of the events, and represents a type of threat that the event poses to Japanese energy supply security. The mapping from events to threats is a form of threat recognition, and requires knowledge of world energy situation in the context of which the event has occurred. Secondly, the policy options may be indexed by the threats.

The state of her foreign relations imposes constraints on Japanese policy options. Thus, even if some policy option were indexed by the threats posed by an energy-related event, Japan might not invoke the policy option in certain states of the world. The constraints may be determined computationally efficiently by *classifying* the relevant world states onto a small number of precompiled constraint types. Each constraint type is an equivalence class of some subset of the states of the world. Now, the threats to Japanese energy supply security posed by an energy-related event and constraints imposed by the state of her foreign relations may combined into *complex indices* for indexing the policy options. The preparation of the complex indices too may be done computationally efficiently by *classifying* the threat types and the constraint types onto precompiled complex indices. Finally, the complex indices may be used to *select* the subset of policy options appropriate for the situation from the set of stored policy options.

4. A Model for Japanese Energy Policy Decision Making

JESSE is an *integrated* knowledge-using problem solving and planning system with *explanation* capabilities. It models Japanese policy decision making in the domain of her energy supply security. Following our analysis above, JESSE contains three classification modules and a module for Plan Selection and Refinement. This is shown in Figure 1. The modules in JESSE communicate with each other *via* a shared memory.

4.1. The Classification Modules

Classification, as we have mentioned earlier, is an elementary generic task [Chandrasekaran, 1986, 1987]. Abstractly, the Classification task is to map a description of some situation onto precompiled concepts in a taxonomy. Hierarchical classification is a strategy for accomplishing the task of Classification computationally efficiently. In hierarchical classification, the precompiled concepts are organized in a taxonomic hierarchy. Associated with each concept in the hierarchy is a knowledge containing, problem solving agent that is sometimes called a specialist for the concept. The control of problem solving is top-down. Each classification agent, when invoked, matches its concept with the situation description. If the match succeeds, then the specialist establishes the concept, and invokes its sub-agents who repeat the process. If the match fails, then the specialist rejects its concept. This control strategy has been called Establish-Refine.

CSRL (for Conceptual Structures Representation Language) is a high level knowledge representation language that embodies the strategy of hierarchical classification [Bylander and Mittal, 1986]. CSRL may be thought of as a generic tool for building a problem solving system for the generic task of Classification, It may be also thought of as a shell; as soon as the domain knowledge is represented in the shell, the language interpreter creates the problem solver. CSRL provides to an expert system designer with an advantage over, say, a rule-based language, similar to the advantage that programming languages provide over assembly languages to the computer programmer. The classification modules in JESSE have been implemented in CSRL.

The first classification module accepts from the user a description of a specific energy-related event, as well data about the world energy situation, and maps it onto threats posed to Japanese energy supply security. The module contains twenty nine threat types organized in a five level taxonomic hierarchy. A portion of the hierarchy is shown in Figure 2. The label EnergyFlow in the figure stands for the threat of increase in the cost of energy, and similarly, ImmediateCost represents an immediate increase in the cost. The label CostDueToChangeInExportCapability represents the threat of increase in the cost of energy *secondary* to a decrease in the flow of energy due to reduced export capability of some energy producing country (or countries).

Associated with each threat type is a classificatory agent. When an agent in the classificatory hierarchy is called by its super-agent, then the agent asks certain questions of the user who may reply by answering "Yes", "No", or "Unknown". The relevant questions are precompiled into the agent. In this way information about a specific energy-related event and the world energy situation is acquired from the user. The agent also contains knowledge in the form of production rules that maps the information acquired from the user onto a confidence value. The confidence value of a threat type is a measure of the likelihood that the event will pose that threat to Japanese energy supply security. CSRL uses an ordinal scale for expressing the likelihood. In this way a likelihood value for the threat type is determined. If the likelihood value is high then the threat is established, otherwise it is rejected. If an agent establishes the corresponding threat type, then it invokes its sub-agents who repeat the process.

The second classification module similarly acquires from the user a description of some specific aspects of Japanese foreign relations, and maps it onto constraints on her policy options. The specific aspects of Japanese foreign relations represented in JESSE are Japanese relations with far east Asian countries, Japanese-US security relations, openness and stability of the international economic order, US support for the international economic order, and access to foreign markets for Japanese exports. The module contains classificatory agents corresponding to sixteen constraint types organized in a three level taxonomic hierarchy.

The third classification module reads from the shared memory the threats posed to Japanese energy supply security by a specific energy-related event as determined by the first classification module, and other constraints imposed on her policy options by her foreign relations as determined by the second classification module, and then maps them onto complex indices for plan selection. The module contains classificatory agents corresponding to seventeen complex indices in a four level taxonomic hierarchy.

4.2. The Module for Plan Selection and Refinement

Plan Selection and Refinement is another elementary generic task. Abstractly, the plan selection and refinement task is to design (typically in association with other tasks) teleological objects such as devices or plans [Brown and Chandrasekaran, 1986]. The object structure is known at some level of abstraction. Concepts corresponding to components of the object are organized in a hierarchy mirroring the object structure. Associated with each concept is a knowledge containing planning agent that is sometimes called a specialist for the concept. Each agent has precompiled plans which can make choices of subcomponents, and may call upon sub-agents for plan refinement. Associated with each plan is a plan sponsor. Each plan sponsor contains knowledge that enables it to determine if its plan is applicable. The control of planning is top-down. Each planning agent, when invoked, calls on its plan sponsors to sponsor applicable plans, and selects the plan that best suits the specifications. The selected plan invokes planning agents at the next lower level in the hierarchy for refinement of the plan. Thus, the control strategy is Select-Refine.

DSPL (Design Specialists Planning Language) is a knowledge representation language that supports Plan Selection and Refinement among other tasks [Brown and Chandrasekaran, 1986]. Like CSRL, DSPL too may be thought of as a generic tool, or as a shell. The module for Plan Selection and Refinement has been implemented in DSPL, which comes with sophisticated explanation capabilities. The module contains nineteen planning agents organized in a three level hierarchy. This is shown in Figure 3.

Each planning agent in the hierarchy is responsible for a precompiled plan, and for each plan there is a plan sponsor. Each plan sponsor contains a table of conditions in the form of production rules for the invocation of the corresponding plan. A plan sponsor, when invoked, reads from the shared memory the values of relevant complex indices as determined by the third classification module. It then matches the values with the conditions in its table, and sponsors the plan if the the match is successful. This process is repeated for each plan in the planning hierarchy starting from AnticipatoryPolicy which is the top level planning agent.

4.3. An Example: The Iranian Revolution

Let us partially trace, in English language for ease of understanding, the policy decision making process of JESSE for the real-world case of the Iranian revolution of 1979. The first classification module establishes that the Iranian revolution poses major and immediate threats to Japanese energy supply security both due to reduced energy flow, and increased energy costs. The basis for this determination is the user supplied information that the energy export capability of Iran will decline, that her energy export policy would change, that Japan imports substantial amount of energy from Iran, and that there is a shortage of energy in the world energy markets.

The second classification module similarly establishes that there are minor problems in Japanese relations with some far east Asian countries, and potential problems with the openness and stability of the international economic order. The third classification module determines that the threat to her energy supply security is the dominant international problem facing Japan, with few constraints on Japanese policy options, and prepares complex indices for plan selection.

The module for plan selection and refinement at the lowest level in the planning hierarchy invokes *only* the plans to buy energy shares at the stock market, to subsidize depletable energy resources, to develop renewable energy resources, to reduce internal demand for energy, to provide incentives for efficient use of energy, to increase stockpiles of energy, to purchase energy from energy exporting countries other than Iran, to induce Iranian dependence on Japanese technology, to bolster other energy exporting countries, and to fund international energy research and development (see Figure 3).

5. Discussion of the Model

There are several aspects to our model of Japanese energy policy decision making in the domain of her energy supply security that deserve special mention.

5.1. Model of Group Cognition

Our model is at the level of aggregation of the Japanese political and economic elite, rather than at the level of a single individual, for instance the Japanese Prime Minister Mr. Noboru Takeshita, or of an organization such as M.I.T.I.

In section 2.2, we argued for the utility of the concept, "group cognition." There are at least two reasons why it is possible to model Japanese energy policy decision making as an instance of group cognition. These two points also serve as reasons why the criteria for applicability of an information pprocessing model, as enumerated in this article's introduction, apply to the domain of Japanese energy policy decision making. In particular, these points speak to the requirement that a general mode of problem solving be identifiable.

1. More than is the case in many other countries, Japanese decision makers have similar political socialization patterns. The preponderance of Japanese civil servants, for instance, have been educated at Tokyo University, with most of the remainder having been educated at Kyoto University. (See [Richardson and Flanagan, 1984, Kubota,

1969].) The self selection process for those who want to be decision makers leads to the study of Law and Economics in quite a high percentage of cases. Generally similar foreign policy world views is hardly a surprising result.

2. In contrast to decision making in many other nation-states, Japanese energy policy decision making is not subsumed by institutional or interagency rivalry. While the common American view of "Japan, Inc." grandly overstates the point, business and government do not have a deep institutional rivalry that diminshes the possibility of acting in consort. (See, for instance, [Samuels, 1987]. This does not mean that all Japanese actions are consistent with each other. (In fact, our model allows and exhibits quite inconsistent Japanese decisions.) It means, however, that decisions are more likely to exhibit a tracable cognitive base. Behavior based on compromises between agencies, that would be quite difficult to capture as group cognition, is less common in Japanese energy policy decision making than in many other nations' foreign policy decision domains.

While these characteristics, in broad form, are not unique to Japan, they raise interesting questions, which we address below, about the scope conditions for generalizing from our model to decision making by political and economic elites of other nation-states.

5.2. Generalizability of the Model

In discussing the issue of generalizability of our model, some of the characteristics of Japanese foreign energy policy decision making become relevant as potential sources of scope conditions for a more comprehensive information processing theory of foreign policy decision making.

1. Since World War II, we argue that Japan has pursued a largely economically-centered as opposed to a largely military-centered foreign policy.

2. Japan is quite dependent on energy imports, and thus energy supply security is a major concern to her.

3. Japan is believed to have prepared policy options in advance for anticipated threats to her energy supply security.

4. Japan is argued here to adopt multiple policy options even when fewer may suffice, where each policy option represents a possible course of action.

Our model, then, is generalizable to other decision making domains that have characteristics similar to the four above. We hope that it will generate insights for still other domains, but it would not, of course, be able to generalize its results directly to such domains.

When considering the issue of generalizability, it is important to note that what we see as the core of the model is the way in which information is processed, and not the substance of the plans in the planning section of the model. In other words, while our vision of progress in science is not in full agreement with [Lakatos, 1970], we see the "hard core" of our theory as the notion and the process of information processing, not as particular plans or actions that the model predicts.

5.3. Validation of the Model

We have been working on validating our model in a number of different ways.

1. We have tested our model for different situations that have actually occurred in the recent past, as evidenced by the example of the Iranian revolution given earlier. Another example of an actual situation for which we have tested our model is the removal of Sheik Yamani from the post of the Oil and Petroleum Minister of Saudi Arabia. Our results show that the performance of JESSE is reasonable. However, we

should add that building a performance system is not our major objective; we are more interested in understanding the process by which national elites arrive at policy decisions.

2. We have tested our model on hypothetical situations also. An example is the hypothetical situation in which Indonesia and Malaysia are at war, and Malaysia has threatened to close the Strait of Malacca to all international shipping.
3. We have demonstrated the system to a few domain experts. This has been an attempt to check the process validity of our model. Their judgment so far has been that the energy policy decision making process followed by JESSE is plausible.
4. We have conducted a literature survey to determine if there is some evidence that Japan actually does follow the energy decision making process modeled by JESSE. Japanese language documents (e.g., M.I.T.I. White Paper) are part of this survey. Since the model itself is based upon interviews with Japanese political and economic elites, we are not checking the model against information from which we built it. Our literature "tests" suggest that Japan indeed does classify energy-related events onto the types of threats that they pose to her energy supply security, and does select and refine stored plans.

While the above tests of our model are clearly empirical in nature, we have chosen not to undertake any quantitative statistical tests. We feel that for empirical validation of a model such as ours, the tests that we have just described are more appropriate than statistical tests. One reason for this conviction is that our model allows for such a broad base of multiple outcomes. In other words Japan, in our model, can undertake no actions in response to an external event, or they could undertake a dozen or more actions, simultaneously, some of which would seem contradictory. Therefore, statistical tests such as those offered by [Bueno de Mesquita, 1981] are inappropriate. We have not simplified our model to look at such dichotomies as "war" or "no war." Instead, our outputs can vary as widely as allying with a previously hostile nation to currying favor through foreign aid or to overtly deciding to take no action. Additionally, each of the four tests outlined above examine both outcome and process validity. Our position is that we offer this model into the academic debate concerning how decisions, including Japanese decisions, are made. The code of the model itself, with annotation, serves as the Appendix to this paper,[4] for the reader's examination. Both our figures and our descriptions of the sample case that we "ran through" the model add to the Appendix to give the reader a base to assess our model. We claim neither that it is the only true model nor that it is the best. We do, however, claim that it illuminates aspects of decision making that other efforts have not done. Over time, you the reader, as part of the academic community of scholars studying decision making, are the ultimate judge of these claims.

5.4. Extensions of the Model

As we see it, JESSE presently stands on it own as a plausible information processing model of Japanese energy decision making. In the future, we hope to even further improve the model. The two directions for further refinement and improvement that we anticipate are as follows:

1. As we have mentioned earlier, one of the tasks that JESSE performs is *reactive planning*. Reactive planning is event driven rather than goal directed; there are no explicitly represented goals in JESSE. We believe that along with reactive planning Japanese energy policy decision making also involves *maintenance planning*. In maintenance planning the goals of maintaining certain functional states in a stationary

[4]Since the Appendix is 59 pages long, we have not attached it to all versions of this paper. If the reader does not find the annotated code appended to this version of the paper, she can obtain one by writing the authors.

state, for instance maintaining the Strait of Hormuz open to international shipping, are explicitly represented. Maintenance planning involves the task of goal identification, which uses a hierarchy of goals. We have developed a preliminary design for maintenance planning.

2. We are working towards augmenting JESSE with a database that allows for knowledge directed data abstraction and inference. The current version of JEESE lacks this capability. Thus, JESSE may acquire from the user knowledge regarding an energy shortage in the world energy markets, and later may need to know if there is an energy glut, but cannot infer it from prior knowledge. An intelligent database would alleviate this problem.

6. Conclusions

When authors from two disciplines undertake research together, their conclusions necessarily address at least two accademic audiences. The conclusions that follow address both political scientists and computer scientists.

At the outset of this paper, we briefly surveyed a number of alternative political science approaches to understand Japanese foreign and energy policy successes. We have now presented a model that, based on our theory of how Japanese elite process information, includes some of the strong points of these alternative approaches. Concepts of neo-Marxists, liberal economists, realists, and other students of foreign policy decision making have been captured when they are reflected in the "thinking" of the Japanese elite.

JESSE is a significant research endeavor, because it has attempted to represent an understanding of decision making without modelling the behavior of specific institutions or of specific individual decision makers. Despite that (and we would argue that it is in fact **because** of that), the information processing approach or metaphor incorporated in JESSE has allowed us to capture Japanese behavior in quite a plausible manner.

On a substantive level, we have captured a great deal of Japanese behavior by representing Japanese group cognition as planning and classifying in a specific order. The classification and the planning have been guided by an economically centered conception of national security. With these assumptions as a base, we have been able to reason through some quite complex decisions. We have also, in effect, operationalized what it means to be guided by an economically centered conception of national security. For the student of foreign policy, the contrast between such economically centered classifications as "energy flow" versus "energy cost" stands in sharp contrast to such traditional militarily centered classifications as "militarily strategic ally" versus "potential military aggressor.

Social metaphors have often been used to understand the structure, the function, and the behavior of the individual human mind. It is still relatively uncommon, however, to use mental metaphors in an attempt to understand the "mind" of organized collectives of humans, such as national political and economic elites. Our work on Japanese energy policy decision making in the domain of her energy supply security is a small step in that direction. We have shown that Japan performs the complex information processing task of constrained decision making which involves the tasks of threat recognition, constraint formulation, complex index preparation, and reactive planning. We have provided a functional architecture for performing these tasks. Thus, JESSE contains multiple classificatory modules that recognize threats, formulate constraints, and prepare complex indices. It contains also a Plan Selection and Refinement module that performs reactive planning. As we described earlier, each classification module is made up of a small number of problem solving agents that cooperate to accomplish their collective task. Similarly, the module for Plan Selection and Refinement contains a small number of cooperating planning agents. Thus, the complex information processing task of decision making is achieved collectively by an ensemble of problem solving and planning agents acting in concert with one another.

From our analysis of Japanese energy policy decision making it appears that a central issue in group cognition, as in individual human cognition, is that of computational complexity of complex information processing tasks that need to be performed. We believe that much of the functional architecture of cognition, individual as well as group, is tuned towards performing complex tasks computationally efficiently with limited computational resources. The computational architecture of the "brain" of national political elites may well allow for more complexity than does the computational architecture of the human brain, but the issues remain the same. In the case of human information processing the issue of the computational complexity often is tackled by decomposing the complex task into a small set of generic tasks. The knowledge organizations and control regimes specific to each constituent generic task are such that that its functionality can be achieved computationally efficiently. Our work suggests that the issues of computational complexity in the case of group cognition also may be amenable to the same approach. It appears to us that the use of knowledge organizations to perform complex tasks efficiently might provide a bridge between our understanding of individual and group cognition. And we can further understand political decision making through this concept of group cognition.

Acknowledgments

This paper has benefited from discussions with Davis Bobrow, David Herman, Charles Hermann, Dwain Mefford, Richard A. Miller, Bradley Richardson, Brian Ripley, Harvey Starr, and Michael Weintraub. We are grateful to them. We especially thank Davis Bobrow, Brian Ripley, and Michael Weintraub for their contributions to the research reported here. We acknowledge the support of the Mershon Center of the Ohio State University, the U.S. Air Force Office of Scientific Research (AFOSR-87-0090), and the U.S. Defense Advanced Research Projects Agency (RADC-F30602-85-C-0010).

References

[Allison, 1971] Allison, Graham T. Essence of Decision, Boston: Little Brown.

[Bobrow *et al.*, 1986] Bobrow, Davis, Donald Sylvan and Brian Ripley. "Japanese Supply Security: A Computational Model". Presented at the Annual Meeting of the *International Studies Association*, Anaheim, March 25-29, 1986.

[Bremer, 1987] Bremer, Stuart A. The GLOBUS Model: Computer Simulation of Worldwide Political and Economic Developments, Boulder: Westview, 1987.

[Brown and Chandrasekaran, 1986] Brown, David and B. Chandrasekaran. "Knowledge and Control for a Mechanical Design Expert System". IEEE Computer, 19(7):92-100, July 1986.

[Bueno de Mesquita, 1981] Bueno de Mesquita, Bruce. The War Trap, New Haven: Yale University Press, 1981.

[Bylander and Mittal, 1986] Bylander, Tom and Sanjay Mittal. "CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling", AI Magazine, 7(3):66-77, August 1986.

[Carbonell, 1981] Carbonell, Jaime. "POLITICS". In Inside Computer Understanding: Five Programs Plus Miniatures, Roger Schank and Christopher Riesbeck (editors). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1981, pages 259-317.

[Carr, 1946]. Carr, Edward Hallett. The Twenty Years Crisis 1919-1939: An Introduction to the Study of International Relations, Second Edition. London: Macmillan, 1946.

[Chandrasekaran, 1986] Chandrasekaran, B. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design". IEEE Expert, 1(3):23-30, Fall 1986.

[Chandrasekaran, 1987] Chandrasekaran, B. "Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks". In the Proceedings of the Tenth *International Joint Conference on Artificial Intelligence*, Milan, Italy August 23-28, 1987, pages 1183-1192.

[Gaucas and Brown, 1987] Gaucas, Dale and Allen Brown. "A Role for Assumption-Based and Nonmonotonic Justifications in Automating Strategic Threat Analysis". In the Proceedings of *Knowledge-based Systems Workshop*, Sponsored by DARPA, St. Louis, Missouri, April 21-23, 1987, pages 158-165.

[Goel and Chandrasekaran, 1987] Goel, Ashok and B. Chandrasekaran. "JESSE: A Conceptual Model of Political Decision Making". Presented at the Annual Meetings of the *Midwest Artificial Intelligence and Cognitive Science Society*, Chicago, April 1987; extended abstract appears in the proceedings, pages 133-135.

[Goel *et al.*, 1987] Goel, Ashok, B. Chandrasekaran and Donald Sylvan. "JESSE: An Information Processing Model of Policy Decision Making". In the Proceedings of the Third Annual *Expert Systems in Government Conference*, Sponsored by IEEE Computer Society, Washington D.C., October 1987, pages 178-187.

[Job, 1987] Job, Brian L., Douglas Johnson, and Eric Selbin. "A Multi-Agent, Script-Based Model of U.S. Foreign Policy Towards Central America". Paper delivered at the Annual Meeting of the *American Political Science Association*, September 3-7, 1987.

[Kubota, 1969] Kubota, Akira. Higher Civil Servants in Post-War Japan: Their Social Origins, Educational Backgrounds, and Career Patterns. Princeton: Princeton University Press, 1969.

[Krasner, 1978] Krasner, Stephen D. Defending the National Interest: Raw Materials Investments and U.S. Foreign Policy. Princeton: Princeton University Press, 1978.

[Lakatos, 1970] Lakatos, Imre. "Falsification and the Methodology of Scientific Research Programmes". In Imre Lakatos and Alan Musgrave (editors), Cnticism and the Growth of Knowledge. Cambridge, England: Cambridge University Press, 1970, pages 91-195.

[Lau and Sears, 1986] Lau, Richard R. and David O. Sears (editors). Political Cognition: The Nineteenth Annual Carnegie Symposium on Cognition. Hillsdale, N.J.: Lawrence Erlbaum, 1986.

[Lenat *et al.*, 1983] Lenat, Douglas, Albert Clarkson, and Garo Kiremidjian. "An Expert System for Indications and Warning Analysis". In the Proceedings of the Eighth *International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, August, 1983.

[Majeski and Sylvan, 1987] Majeski, Stephen, and David J. Sylvan. "A Constitutive Model of the U.S. Foreign Policy Recommendation Process". Paper presented to the Annual Meetings of the *American Political Science Association*, 1987.

[Meadows *et al.*, 1982] Meadows, Donella, J. Richardson, and G. Bruckman. Groping in the Dark: The First Decade of Global Modeling. New York: John Wiley and Sons, 1982.

[Mefford, 1987] Mefford, Dwain. "The Policy Process and the Evolution of Argument Casting Theory in the Form of a Series of Computer Programs". Presented at the Annual Meetings of the *International Studies Association*, Washington D. C., Apnl 15-18, 1987.

[Morgenthau, 1966] Morgenthau, Hans J. Politics Among Nations Fourth Edition. New York: Knopf, 1966.

[Richardson and Flanagan, 1984] Richardson, Bradley M., and Scott C. Flanagan. Politics in Japan. Boston: Little Brown, 1984.

[Samuels, 1987] Samuels, Richard J. The Business of the Japanese State: Energy Markets in Comparative and Historical Perspective. Ithaca: Cornell University Press, 1987.

[Shapiro and Bonham, 1982] Shapiro, Michael J. and G. Matthew Bonham, "A Cognitive Process Approach to Collective Decision Making". In Christer Johsson (editor), Cognative Dynamics and International Politics. New York: St. Martin's, 1982.

[Simon, 1985] Simon, Herbert. "Human Nature in Politics: The Dialogue of Psychology with Political Science". American Political Science Review, 79(2):293-304, June 1985.

[Sylvan, 1987] Sylvan, Donald. "Supplementing Global Models with Computational Models: An Assessment and An Energy Example". Behavioral Science, 32(3):212-231, July 1987.

[Sylvan *et al.*, 1987] Sylvan, Donald, Davis Bobrow, and Brian Ripley. "Economic Security as Managed Interdependence: A Computational Model of Japanese Energy Policy". Presented at the Annual Meetings of the *International Society of Political Psychology*, San Francisco, July 4-7, 1987

[Thorson, 1984] Thorson, Stuart J. "Intentional Inferencing in Foreign Policy: An AI Approach". In Donald A. Sylvan and Steven Chan (editors), Foreign Policy Decision Making: Perception, Cognition, and Artificial Intelligence. New York: Praeger, 1984.

[Wallerstein, 1984] Wallerstein, Immanuel M. The Politics of the World Economy: the States, the Movements, and the Civilizations, New York: Cambridge University Press, 1984.

Understanding Device Feedback

Ashok Goel
B. Chandrasekaran

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus
Ohio 43210

(614)-292-1413
goel-a@ohio-state.arpa

Submitted to AAAI-88
Primary Topic: Knowledge Representation
Secondary Topic: Commonsense Reasoning(Qualitative Reasoning)
Length: Approximately 4000 words; this includes estimates for figures.

Abstract

Reasoning about the behaviors of a device requires, of course, a language for representing the reasoner's understanding of the device. Moreover, reasoning about complex devices computationally efficiently requires a scheme for organizing the reasoner's knowledge of the device behaviors such that they are easily accessible at the needed level of abstraction. In the *functional representation scheme* [5] for expressing a problem solving agent's understanding of a device, the behaviors are organized around the functions of the device and its structural components. In this paper we extend this scheme to express an agent's understanding of feedback and feedforward interactions common in complex devices. We discuss how feedback and feedforward functions lead to nonlinear device behaviors, and the knowledge structures needed to capture these functions and behaviors.

Understanding Device Feedback

Ashok Goel and B. Chandrasekaran

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

1. Functional Representation

Most research on qualitative reasoning has been focused on predicting and explaining behaviors of physical devices and processes (*e.g.* [3]). Reasoning about the behaviors of a device requires, of course, a language for representing the reasoner's understanding of the device. Moreover, reasoning about complex devices computationally efficiently requires a scheme for organizing the reasoner's knowledge of the device behaviors such that they are easily accessible at the needed level of abstraction. In relation to this, Sembugamoorthy and Chandrasekaran [5] have proposed that a problem solving agent's knowledge of device behaviors may be organized around higher level abstractions such as the functions of the device and its structural components. In their *functional representation scheme* an agent's understanding of a device is expressed as hierarchically organized schemata, in which the nodes are the intrinsic functions of the device and its components, and the arcs are the behaviors that result in the accomplishment of these functions. The behaviors themselves are represented as acyclic directed graphs, in which the vertices are partial states of the device, and the edges are causal state transitions.

The central thesis of this scheme is that problem solving agents often understand the functioning of a complex device by decomposing the device function into the functions of its structural components. The functioning of a component is similarly understood in terms of the functions of its subcomponents. This decomposition may go on upto as many levels as needed, with only limited interactions between a few components at any level. In the recomposition phase, the functions of the components are composed by behaviors to obtain the function of the device. The function of a device component is similarly obtained by behaviors that compose the functions of its subcomponents. The specification of a behavior at any level may include pointers to deeper knowledge and assumptions underlying the recomposition at that level.

The functional representation scheme has been used for constructing deep models of how problem solving agents understand causal phenomena such as the functioning of simple physical devices [5] and the behaviors of plans viewed as abstract devices [1]. These deep models in turn have been used for qualitative reasoning about the functions and behaviors of various devices, most extensively in the diagnosis of malfunctioning devices [2]. Our aim in this paper is to extend the functional representation scheme to express a problem solving agent's understanding of feedback and feedforward interactions common in complex devices. We will discuss how feedback and feedforward functions lead to nonlinear device behaviors, and the knowledge structures needed to capture these functions and behaviors

2. Structure of Feedback

Let us consider the Nitric Acid Cooler (NAC), a device commonly used in chemical processing plants, for illustrating feedback and feedforward interactions. The mechanical circuit for (a simplified version of) NAC is shown schematically in Figure 1. Hot Nitric Acid (HNO_3) enters the cooler at p_1 with flow rate R and temperature T_1, and exits at p_4 with the same flow rate and a lower temperature T_2, where p_1, p_2 ... are points in the device space. Similarly, cold water (H_2O) is pumped into the cooler at p_5 with flow rate r_1 and temperature t_1, and exits at p_8 with flow rate r_2 and a higher temperature t_2. Inside the heat exchange chamber heat is transferred from hot Nitric Acid to cold water, thereby cooling Nitric Acid from T_1 to T_2 and heating water from t_1 to t_2. The flow rate R of the inflowing Nitric Acid is measured by a flow sensor, and information about perturbations in its value is communicated to the water pump by a signal c_1 in the wire connecting the sensor and the pump. The pump regulates the rate r_1 at which water flows into the cooler to reflect the perturbations in value of R. This is an example of feedforward control since it is applied before the exchange of heat. Similarly, the temperature T_2 of outflowing Nitric Acid is measured by a temperature sensor, and information about perturbations in its value is communicated to the valve by a signal c_2 in the wire connecting the sensor and the valve. The valve regulates the rate r_2 at which water enters the heat exchange chamber to reflect the perturbations in the value of T_2, and releases excess water. This is an example of feedback control.

We will not devote much space here to the issue of representation of structure except to say that the functional representation language provides primitives for specifying the device components, the relations between them, and their (device independent) functional abstractions. For instance the schema for the structure of NAC would specify that the chamber $\{p_7, p_3, p_8, p_2\}$ is a component of NAC, that the space enclosed by the chamber includes the space enclosed by the pipe $\{p_2 p_3\}$, and that the functions of the chamber are to contain fluid and transport fluid.

3. Function of Feedback (or Interacting Functions)

The top level decomposition of the functions, in terms of which a problem solving agent may understand the functions of NAC is shown in Figure 2. CoolNitricAcidToT_2 is the primary function of the device, where T_2 is some constant temperature. HeatWater is the secondary function of the device: it is also a side function of CoolNitricAcidToT_2. This captures an agent's understanding that while the intended function of NAC is to cool Nitric Acid, as a side effect of this, water is heated as well. Further, while the intention is to keep the temperature T_2 of outflowing Nitric Acid as steady as possible, the temperature t_2 of outflowing water may vary.

At the next level in the network of Figure 2, SupplyWaterToChamberAtRater_2 is a subfunction (or constituent function) of HeatWater; it is also a supporting function for CoolNitricAcidToT_2, *i.e.* its function is to satisfy the preconditions for the accomplishment of the CoolNitricAcidToT_2 function. Similarly, SupplyNitricAcidToPipeInChamber is a subfunction of CoolNitricAcid and a supporting function of HeatWater. This captures an agent's understanding of the *interaction* between the functions of CoolNitricAcidToT_2 and HeatWater, allowing him to reason that since the subfunction for CoolNitricAcidToT_2 is a supporting function of HeatWater and *vice versa*, the Nitric Acid will get cooled if, and only if, water simultaneously gets heated. Further this enables the agent to view the role of functions from *multiple perspectives*: SupplyWaterToChamberAtRater_2 is a subfunction from the perspective of achieving HeatWater, but a supporting function from the perspective of accomplishing CoolNitricAcidToT_2.

At the next lower level, the feedback and feedforward functions of

ControlWaterFlowIntoChamber and ControlWaterFlowIntoCooler are similarly understood as supporting functions for SupplyWaterToChamberAtRater_2. Thus, the feedforward and feedback functions are viewed as fulfilling the preconditions for the accomplishment of some higher level function, in this case the SupplyWaterToChamberAtRater_2 function which is itself a supporting function of CoolNitricAcidToT_2.

The schemas for some of these functions are shown in Figure 3. The underlined expressions are the primitives of a functional representation language, each with an associated semantics. The primitives <u>Given</u> and <u>ToMake</u> provide an input-output specification of the functions while <u>By</u> specifies the behavior that results in the accomplishment of the function. Thus each function in the network can be used to *index* the behaviors responsible for accomplishing it. <u>Provided</u> specifies the states of the device in which only a given function can be accomplished, and relates the function to its supporting

4. Behavior of Feedback (or Nonlinear Behaviors)

The directed graphs for the behaviors that achieve some of the NAC functions discussed above are shown in Figure 3. The primitive <u>Using-Function</u> specifies the function of some component that is used by the behavior in accomplishing some higher level function, while <u>By</u> refers to some lower level behavior. The specification of a behavior may include pointers to deeper causal knowledge and assumptions underlying a causal state transition in the behavior. For instance, Behavior1 for accomplishing the function of CoolNitricAcidToT_2 uses (Generic-Knowledge1) that may be stated as follows: "In accordance with the Zeroth Law of Thermodynamics, in the context of the Chamber(p_1, p_2, p_3, p_4) enclosing the Pipe(p_2, p_3), heat will flow from hot Nitric Acid to cold water resulting in a decrease in the temperature of Nitric Acid from T_1 to some T_2 and an increase in the temperature of water from t_1 to some t_2.

Similarly, Behavior1 accomplishes the CoolNitricAcidToT_2 under Assumption1 which may be stated as follows: The relation between temperature T_1 and flow rate R of inflowing Nitric acid, the desired temperature T_2 of outflowing Nitric acid, and the temperature t_1 and flow rate r_2 of water flowing into the heat exchange chamber, is such that the capacity of water to absorb heat in the chamber exceeds the capacity of Nitric Acid to release heat. In essence, the assumption is that the perturbations in the values of the variables T_1 and R are small enough that it is possible to compensate for them by changing the value of the parameter r_2.

The interactions between the functions of a device are, of course, reflected in the behaviors that accomplish the functions. For instance, Behavior1 for accomplishing the function of CoolNitricAcidToT_2, and Behavior2 for achieving HeatWater shown in Figure 3, interact in that Behavior1 will result in cooling Nitric Acid to T_2 if and only if Behavior2 simultaneously results in heating water. This interaction is being captured by the primitive Predicate which specifies that the causal transition from one device state to another in some behavior is conditional on some other device state being true.

We note that the behaviors for accomplishing these interacting device functions are *nonlinear* in the same sense that the plans to achieve interacting goals are often nonlinear [4]. That is, while the device behaviors can be *partially ordered*, each individual behavior being a linear sequence of causal state transitions, a total ordering of the behaviors is typically not possible. Instead, a network of behaviors mirroring the network of Figure 2 collectively results in the functioning of the device. In fact, for the specific case of the skeletal NAC, the device behaviors are inherently *non-serializable*. Thus, if a problem solving agent were to perform a qualitative simulation to verify whether Behavior1 will indeed lead to cooling of Nitric Acid to T_2, then he will have to perform "in parallel" a simulation to check if Behavior2 indeed results in heating water.

5. Understanding Feedback

In our approach, feedback and feedforward are represented as functions that control the values of certain parameters of the device. These control functions are achieved by nonlinear behaviors that communicate information about perturbations in the values of the device variables. The important point here, however, is that reasoning about the functions and behaviors of a complex device can be computationally very expensive, especially in the presence of feedback and feedforward interactions. It is computationally advantageous to organize the understanding of the device into a hierarchical network of functions such that there are only limited interactions between a few functions at any level. During problem solving, when needed these functions can be used to index the individually linear behaviors responsible for accomplishing them.

Representations of devices are there, of course, to be *used*. In fact their use provides the only criterion for judging their adequacy. We have so far used the functional representation of devices primarily for solving two types of problems. In one, when the diagnostic reasoner has incomplete knowledge of certain types, the functional representation can be interpreted and the missing diagnostic knowledge can often be derived. Since the function of the device is represented as being achieved by means of a behavioral sequence whose causal transitions are ultimately related to the functions of the components, the functional representation yields malfunction hierarchies. Further, since the causal sequences incorporate information about what states fail to result due to malfunctioning of certain components, the representation can also yield observations which may be used to verify malfunction hypotheses. Sticklen [6] has used this idea to develop a diagnostic system which accesses the functional representation of disease processes for deriving additional diagnostic knowledge.

Another use of the functional representation of a device is to derive qualitative simulations, not from first principles or by using qualitative physics, but by tracing the causal paths organized by functions. Sticklen [6] has studied the use of such simulations for examining certain types of interaction between the components of a device. Since the causal sequences are available in stored form and organized functionally, the real work in such simulations is not in the generation of behaviors, but in tracing the effect of certain actions on the functionality of the system.

What functions ought to be included in the representation depends, of course, on the level at which the agent is engaged in problem solving. For instance, if the task is to predict the behavior of a chemical processing plant of which NAC is but one small component, then it is useless to represent the feedback interactions inside NAC. At this level, NAC may best be viewed as a "black box" that operates as a homeostatic device and cools Nitric Acid to a constant temperature. Alternatively, if the task was to explain the functioning of the temperature sensor, then again, it is meaningless to represent feedback interactions at the level of NAC. However, if the task was, say, diagnosis of NAC itself, then a representation of feedback interactions in NAC would be clearly useful. We may add that although we have used NAC as an example to illustrate feedback and feedforward interactions, the functional representation scheme and language that we have used for representing these interactions are device and domain independent and more generally applicable.

6. Functional Representation and Qualitative Simulation

Qualitative simulation is an alternative approach to reasoning about devices in general, and device feedback in particular. In the qualitative simulation method of de Kleer and Brown [3], first the relevant parameters and constraints of the device are determined from its structure and represented as qualitative differential constraints, then a differential perturbation is introduced into the system and a qualitative simulation is performed, and finally changes in the values of the parameters are tracked. There is no *explicit* representation of behavior or function *per se*, instead the changes in the values of the parameters are first interpreted as behaviors which may then be ascribed a function. de Kleer and Brown have illustrated the use of this method for reasoning about device feedback in an air pressure regulator.

There are several features in common to the method of de Kleer and Brown and our scheme for reasoning about device feedback. Both approaches view feedback as a function, not as a behavior. More importantly, there is a major emphasis in both approaches on making explicit the (otherwise tacit) assumptions underlying reasoning about devices. There are clearly several differences between the two approaches as well. While their work is more concerned with the *qualitative physics* of device feedback, our primary concern is with a problem solving agent's *cognition* of feedback. Moreover, while their approach is more concerned with the *correctness* of solutions, we are more concerned with the *computational efficiency* of reasoning.

Given a device structure, there is the task of deriving its behavior, which is the problem that is attacked by qualitative simulation. However the agent also needs to organize this behavior in such a way as to explain how the functions of the device are made possible. For simple systems, the distinction between behavior and function is not significant, since relevant behaviors are often also the functions. For complex systems, however, the functions need to be used to index and organize the causal sequences that the structure-to-behavior reasoning has generated. Thus, the functional representation scheme and the qualitative simulation methodology are best viewed as complementary to each other. While the functional representation scheme seeks to capture the *content* of a problem solving agent's understanding of device feedback, the method of qualitative simulation may provide one of the *mechanisms* by which the agent acquires the representation. This relationship between the two approaches works in the other direction as well. For instance, a major drawback of the method of qualitative simulation is that since simulation is global reasoning process, for complex devices the method can be computationally very expensive, especially in the presence of feedback and feedforward interactions. The functional representation scheme, because of its hierarchical nature, may help localize the qualitative simulation to some portion of the device. The integration of the two approaches to form a complete and coherent framework of how problem solving agents understand the functioning of devices, acquire this understanding, and use it for problem solving, however, remains an open research issue.

Acknowledgments

This research has been supported by the Defense Advanced Research Projects Agency, contract RADC F30602-85-C-0010, the Air Force Office of Scientific Research, grant AFOSR-87-0090, the National Heart Lung and Blood Institute, grant HL-38776, and the National Science Foundation, grant CBT-8703745.

References

[1] B. Chandrasekaran, J. Josephson and A. Keuneke. "Functional Representation as a Basis for Generating Explanations". In the *Proceedings of the IEEE Conference on Systems. Man. and Cybernetics*, Atlanta, Georgia, 1986. pages 726-731.

[2] B. Chandrasekaran, J. Smith and J. Sticklen. "Deep Models and their Relation to Diagnosis". In *Artificial Intelligence in Medicine*, Furukawa (editor), Amsterdam, Netherlands: Science Publishers, 1988 (in press).

[3] J. de Kleer and J. Brown. "A Qualitative Physics Based on Confluences". *Artificial Intelligence*, 24(1984):7-83.

[4] E. Sacerdoti. "The Nonlinear Nature of Plans". In the *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, USSR. 1975. pages 206-214.

[5] V. Sembugamoorthy and B. Chandrasekaran. "Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems". In *Experience, Memory, and Reasoning*, J. Kolodner and C. Riesbeck (editors), Hillsdale, New Jersey: Lawrence Erlbaum, 1986, pages 47-73.

[6] J. Sticklen. "MDX2: An Integrated Medical Diagnostic System". PhD. Dissertation, Department of Computer and Information Science. The Ohio State University, 1987.

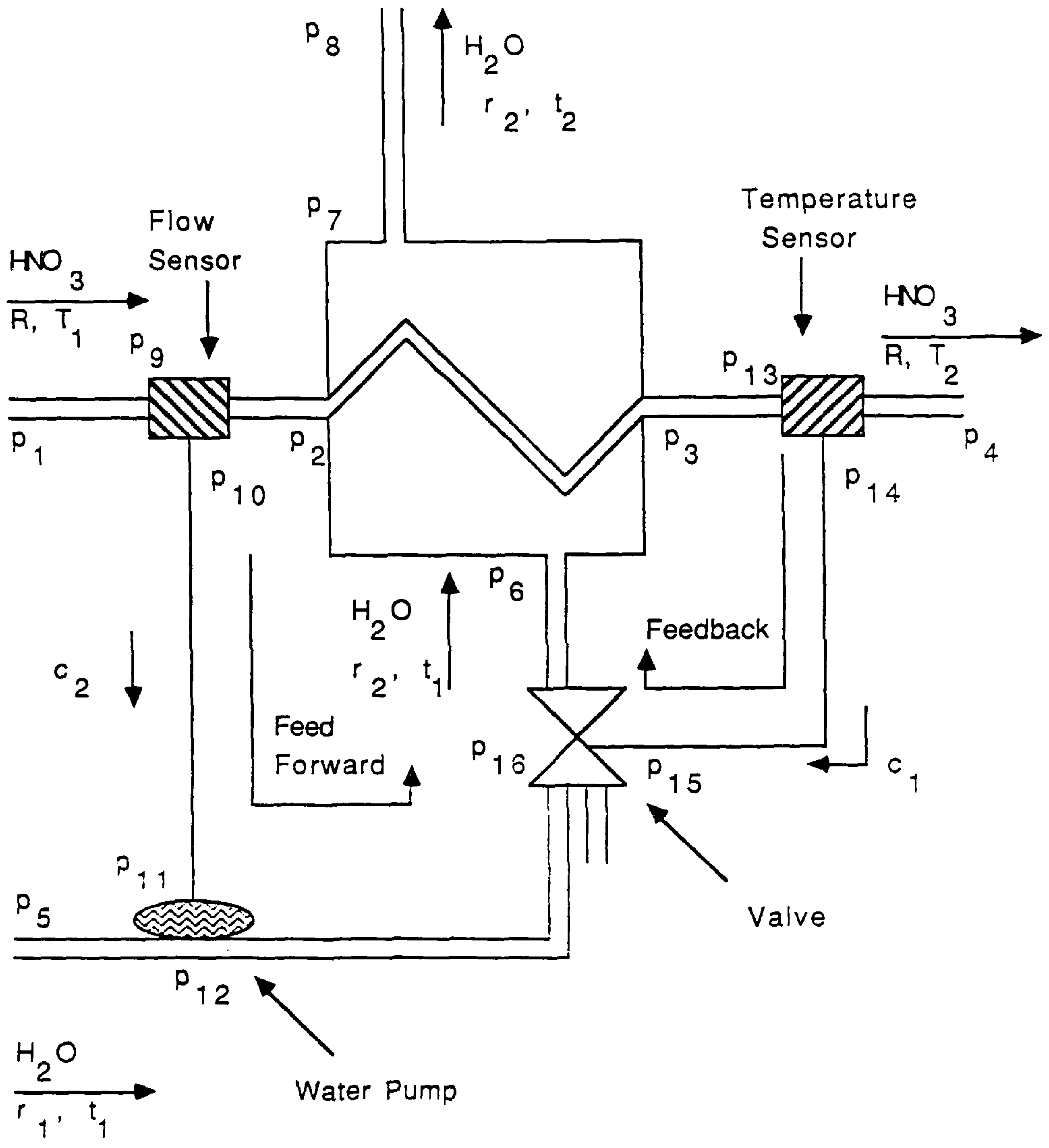


Figure 1 : The Nitric Acid Cooler

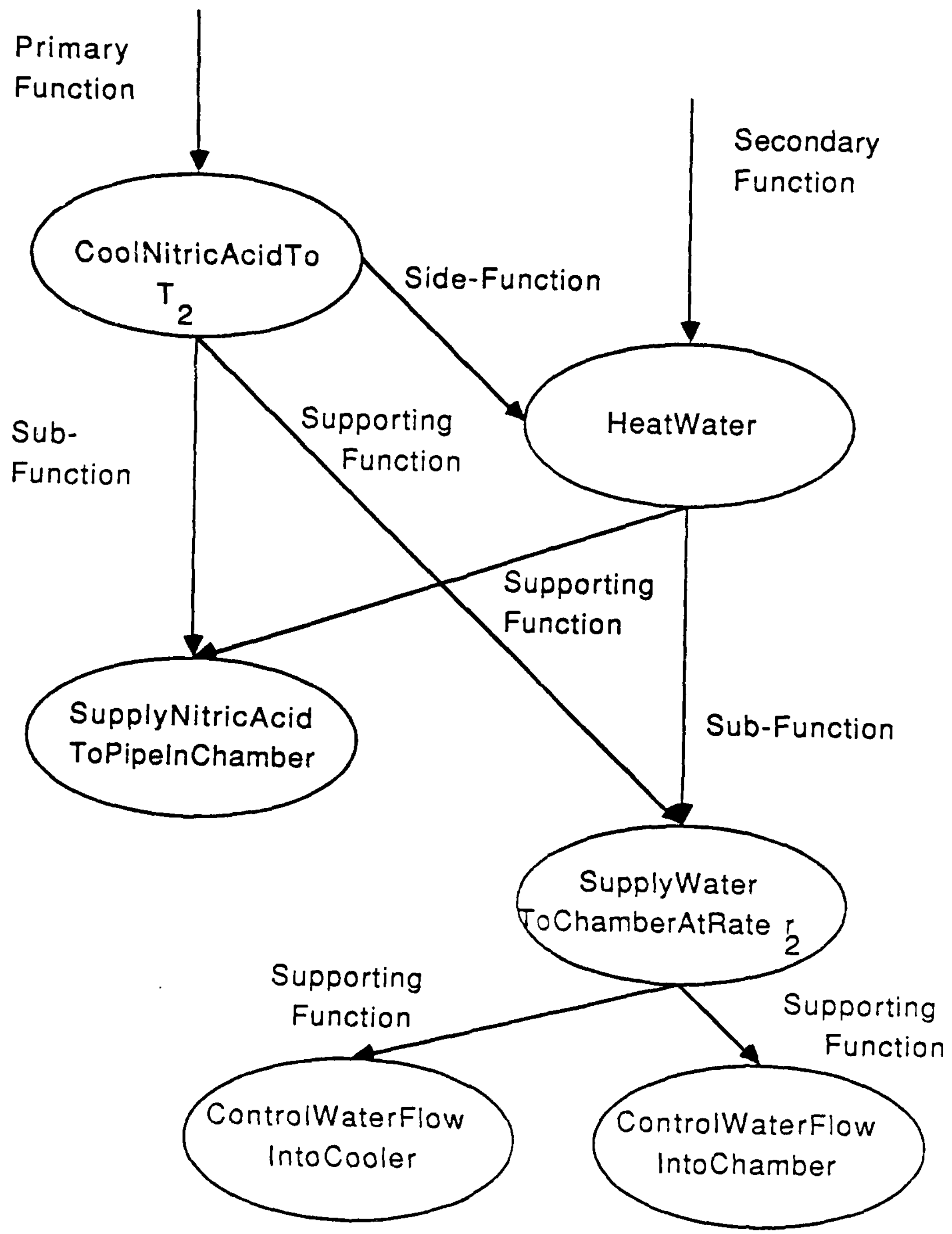


Figure 2 : Functional Organization of the Nitric Acid Cooler

```
Function: CoolNitricAcidTo$T_2$
    Given:
      $HNO_3$ at $p_1$ with
      Flow Rate $R$ and Temperature $T_1$
    ToMake:
      $HNO_3$ at $p_4$ with
      Flow Rate $R$ and Temperature $T_2$
    By: Behavior1
    Provided:
      $H_2O$ at $p_6$ with
      Flow Rate $r_2$ and Temperature $t_1$
End Function CoolNitricAcidTo$T_2$

Behavior1
ToAchieveFunction: CoolNitricAcidTo$T_2$

    $HNO_3$ at $p_1$
    with Flow Rate $R$ and Temperature $T_1$
                !
                ! By: Behavior3
                V
    $HNO_3$ at $p_2$
    with Flow Rate $R$ and Temperature $T_1$
                !
                ! Predicate: $H_2O$ at $p_7$ with
                ! Flow Rate $r_2$ and Temperature $t_1$
                !
                ! As-Per: Generic-Knowledge1
                !
                ! With: Assumption1
                !
                ! Using-Function: Transport Fluid
                ! of Pipe $\{p_2,p_3\}$
                V
    $HNO_3$ at $p_3$
    with Flow Rate $R$ and Temperature $T_2$
                !
                ! Using-Function: Transport Fluid
                ! of Pipe $\{p_3,p_4\}$
                V
    $HNO_3$ at $p_4$
    with Flow Rate $R$ and Temperature $T_2$
End Behavior1
```

Figure 3: Some Functions and Behaviors of NAC

```
Function: HeatWater
  Given:
    $H_2O$ at $p_5$ with Temperature $t_1$
  ToMake:
    $H_2O$ at $p_8$ with
    Flow Rate $r_2$ and at Temperature $t_2$
  By: Behavior2
  Provided:
    $HNO_3$ at $p_2$ with
    Flow Rate $R$ and Temperature $T_1$
End Function HeatWater


Behavior2
ToAchieveFunction: HeatWater

  $H_2O$ at $p_5$ at Temperature $t_1$
            !
            ! By: Behavior4
            V
  $H_2O$ at $p_7$
  with Flow Rate $r_2$ and Temperature $t_1$
            !
            ! Predicate: $HNO_3$ at $p_2$ with
            ! Flow Rate $R$ and at Temperature $T_1$
            !
            ! As-Per: Generic-Knowledge1
            !
            ! Using-Function: Transport Fluid of
            ! Chamber{$p_7, p_3, p_8, p_2$}
            V
  $H_2O$ at $p_7$
  with Flow Rate $r$ and Temperature $t_2$
            !
            ! Using-Function: Transport Fluid of
            ! Pipe {$p_7, p_3$}
            V
  $H_2O$ at $p_3$
  with Flow Rate $r$ and Temperature $t_2$
End Behavior2
```

Figure 3(continued): Some Functions and Behaviors of NAC

```
Function: SupplyWaterToChamberAtRate$r_2$
   Given: $H_2O$ at $p_5$ at Temperature $t_1$
   ToMake: $H_2O$ at $p_7$ with
            Flow Rate $r_2$ and Temperature $t_1$
   By: Behavior4
   Provided:
      (i) Control Signal $c_1$ at $p_{11}$
      (ii) Control Signal $c_2$ at $p_{15}$
End Function SupplyWaterToChamberAtRate$r_2$


Behavior4
ToAchieveFunction: SupplyWaterToChamberAtRate$r_2$

   $H_2O$ at $p_5$ at Temperature $t_1$
             !
             ! Predicate: Control Signal $c_1$
             !  at $p_{16}$
             !
             ! Using-Function: Pump $H_2O$ of
             ! WaterPump
             V
   $H_2O$ at $p_{12}$ with
   Flow Rate $r_1$ and Temperature $t_1$
             !
             ! Using-Function: Transport Fluid
             ! of Pipe {$p_{12}$,$p_{15}$}
             V
   $H_2O$ at $p_{15}$ with
   Flow Rate $r_1$ at Temperature $t_1$
             !
             ! Predicate: Control Signal $c_2$
             ! at $p_{15}$
             !
             ! By: Behavior7
             V
   $H_2O$ at $p_7$ with
   Flow Rate $r_2$ at Temperature $t_1$
End Behavior4
```

Figure 3(continued): Some Functions and Behaviors of NAC

```
Function: ControlWaterFlowIntoChamber
  Given: $HNO_3$ at $p_{13}$
         with Flow Rate $R$ and Temperature $T_2$
   ToMake: Control Signal $c_2$ at $p_{15}$
   By: Behavior6
End Function ControlWaterFlowIntoChamber


Behavior6
ToAchieveFunction: ControlWaterFlowIntoChamber

   $HNO_3$ at $p_{13}$
   with Flow Rate $R$ and at Temperature $T_2$
         !
         ! Using-Function: Measure Temperature
         ! of Temperature Sensor
         V
   Control Signal $c_2$ at $p_{14}$
         !
         ! Using-Function: Transmit Signal of
         ! Wire $\{p_{14}, p_{15}\}$
         V
   Control Signal $c_2$ at $p_{15}$
End Behavior6
```

Figure 3(continued): Some Functions and Behaviors of NAC

CONNECTIONISM AND INFORMATION PROCESSING ABSTRACTIONS

B. CHANDRASEKARAN
A. GOEL AND D. ALLEMANG
88-BC-CONPROAB

Connectionism and Information Processing Abstractions

B. Chandrasekaran, Ashok Goel, and Dean Allemang

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

January, 1988

This paper is to appear in the proceedings of the AAAI spring symposium on Parallel Models of Intelligence: How Can Slow Components Think So Fast?, at Stanford University, in March 1988. A shorter version of the paper is to appear as a commentary on Smolensky's paper "On the Proper Treatment of Connectionism" in the Journal of Brain and Behavior Scienes, 1988.

Table of Contents

Connectionism and Information Processing Abstractions: The Message Still Counts More Than the Medium

B. Chandrasekaran, Ashok Goel, and Dean Allemang

Abstract

Since Connectionism challenges some of the basic assumptions on which much of Artificial Intelligence research has been based, it is important to examine the nature of representations and the differences between the Symbolic and Connectionist paradigms in this regard. Even though Symbolic and Connectionist systems may appear to yield the same functionality, we discuss how there is greater distinction between them than the Connectionist architectures being mere implementations of corresponding Symbolic algorithms. The two accounts differ fundamentally in terms of representational commitments, and thus in principle they offer alternative information processing theories. Nevertheless, we argue that the hard work of theory formation in Artificial Intelligence remains at the level of proposing the right information processing abstractions since they provide the *content* of the representations. When, and if, we have Connectionist implementations solving a variety of higher level cognitive problems, the design of such systems will have these information processing abstractions in common with the corresponding Symbolic implementations. The information processing level specification of a theory of intelligence will then lead to decisions about which transformations on representations are best performed by means of Symbolic algorithms and which by Connectionist networks. In essence we claim that while Connectionism is a useful corrective to some of the basic assumptions of the Symbolic paradigm, for most of the central issues of *intelligence* Connectionism is only marginally relevant.

1. Introduction

Much of the theoretical and empirical research in Artificial Intelligence (AI) over the past thirty years has been based on the so-called "Symbolic" paradigm --- the thesis that algorithmic processes that interpret discrete symbol systems provide a good basis for understanding intelligence. It is for this reason that AI is so closely associated with Computer Science. In spite of what we regard as significant achievements of AI in beginning to provide a computational language to talk about the nature of intelligence, there have been recurring doubts about the Symbolic paradigm. In addition to the earlier neural net modellers and the perceptron theorists we now have the modern connectionists who offer largely analog processes implemented by weights of connections in a network as a basis for modeling human cognition and perception --- the so-called "Connectionist" paradigm. The not so well-kept secret in AI is that AI internally is in a paradigmatic mess. There is really no broad agreement on the essential nature or formal basis of intelligence. and the proper framework for studying it.

We believe that both Symbolic and Connectionist theories carry a large amount of unanalyzed assumptional baggage. In this paper we examine the features. assumptions, and the claims of Connectionism. Our aim is to give a broad-brush account of the Connectionist theories of the nature of intelligence. Such broad-brush accounts, by their very nature, tend to treat things a little too neatly. Nevertheless, we believe that a treatment in such broad terms is necessary to make sense of a field such as AI which is in conceptual confusion about its foundations.

2. Characterization of the Issues

2.1. AI as a Science of Intelligence

Let us make a useful distinction which might eliminate at least some of the arguments about AI: the distinction between "intelligence" and "mind." Many early discussions on the philosophical implications of AI equated the question, "Can machines be intelligent?" with "Are minds machines?". There is a useful alternative to this equation of mind and intelligence, *viz.*, that intelligence is a *tool* of the mind. In fact, there is a tradition in the Eastern philosophies which embodies precisely such a distinction: it views intelligence as an internal sense organ much as sight is an external sense organ. As a sense organ, intelligence interprets the world and makes the information available to the "watcher". Our aim in making this distinction here is not to stake an ultimate position about the irreducibility of mind to mechanism, but merely to remove from the discussion some elements about which AI as a technical discipline has nothing to say at this time. Even the most rabid mechanist within the AI community will need to admit that while AI may have impressively useful things to say about cognition and perception, it simply has nothing technical --- at this time --- to say about consciousness, will, feelings, etc. Thus, we want to take intelligence, and not mind, as the current subject matter of AI.

2.2. *Intelligence as Information Processing on Representations*

While there are theoretical differences between those who subscribe variously to the Symbolic and to the Connectionist paradigms, there also is something that is shared ***almost universally*** among researchers in AI: "Significant (all?) aspects of cognition and perception are best understood/modeled as *information processing activities on representations.*" This description of intelligence does not, however, characterize the class of intelligent processes well enough within the class of all information processing activities. Is there something that can be recognized as the *essential* nature of intelligence that can be used to characterize all its manifestations: human, alpha-centaurian, and artificial?

It is possible that intelligence is merely a somewhat random collection of information processing transformations acquired over eons of evolution, but in that case there can hardly be an interesting science of it. It is also possible that while there may well be interesting characterizations of human intellectual processes, they need not be taken to apply to other forms of intelligence, in which case there need not be anything that particularly restricts attempts to make intelligent machines. While in some sense it seems right to say that human intellectual processes do not bound the possibilities for intelligence, nevertheless, we believe that there is an internal conceptual coherence to the class of information processing activities characterizing intelligence. The oft-stated dichotomy between the simulation of human cognition versus making machines intelligent is a temporarily useful distinction, but its implication that we are talking about two very different phenomena is, we believe, incorrect. In any case, *a* task of AI as a science is to explain human intelligence. The underlying unity that we are seeking can be further characterized by asking, "What is it that unites an Einstein, a man on the street in a Western culture, and a tribesman in a primitive culture, as information processing agents?"

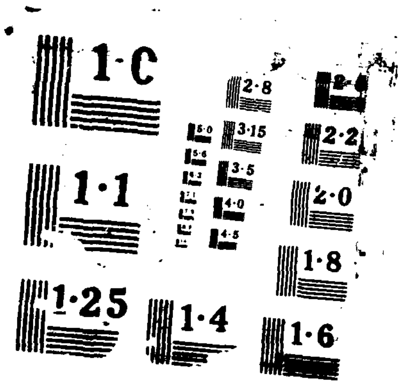
2.3. *The Symbolic and the Connectionist Paradigms*

We have called the thesis that intelligence can be understood by [illegible] processes which interpret discrete symbol systems the Symbolic [illegible]. Stronger versions of the Symbolic paradigm have been proposed by [illegible] as the physical symbol system hypothesis, and elaborated by [illegible] thesis that Symbolic Computationalism is not simply a [illegible] talk about cognition, but that cognition literally is [illegible] tems. It is important to note that this thesis does not [illegible]

[1]This is not a completely satisfactory term [illegible] crete symbolic systems, since we believe that [illegible] wouldn't be representations. However there [illegible] this commitment. Dennett [illegible] Pylyshyn (1987) use the term [illegible] use the term Symbolic [illegible] tion over discrete symbol [illegible]

AD-A193 107 DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS B CHANDRASEKARAN 08 APR 88 2/2

UNCLASSIFIED AFOSR-TR-88-0572 AFOSR-87-0090 F/G 12/9 NL


1·0
2·8
2·5
3·15
2·2
3·5
4·0
2·0
4·5
1·1
1·8
1·25
1·4
1·6

tical sufficiency of current von Neuman computer architectures for the task of understanding intelligence, or a restriction to serial computation. Often disagreements with the Symbolic Computationalism turn out to be arguments for computer architectures that support some form of parallel and distributed processing rather than arguments against computations on discrete symbolic representations.

Let us call the alternative to this the "Non-Symbolic" paradigm, for lack of a better term. Connectionism is an example of this alternative, though not the only one. Connectionism offers to model human cognition and perception by largely analog processes implemented by weights of connections in a network of processing units as we stated earlier. We will provide a more detailed description of the Connectionist framework a little later (see section *4*).

3. The Nature of Representations: Roots of the Debate

3.1. Representational vs. Non-Representational Theories

The Symbolic *vs.* Connectionist debate in AI today is but the latest version of a fairly classic contention between two sets of intuitions each leading to a *weltanschauung* about the nature of intelligence. The debate can be traced at least as far back as Descartes in modern times (and to Plato if one wants to go further back), and the mind-brain dualism that goes by the name of Cartesianism. In the Cartesian world view, the phenomena of mind are exemplified by language and thought. These phenomena may be implemented by the brain, but are seen to have a constituent structure in their own terms and can be studied abstractly. Symbolic logic and other symbolic representations have often been advanced as the appropriate tools for studying these phenomena.

Functionalism in philosophy, information processing theories in psychology, and the Symbolic paradigm in AI all share these assumptions. While most of the intuitions that drive this point of view arise from a study of cognitive phenomena, the thesis is often extended to include perception, *e.g.* in Bruner's (1957) thesis that ***perception is inference.*** In its modern version the Cartesian viewpoint appeals to the Turing-Church hypothesis as providing a justification for limiting attention to Symbolic Computational models. These models ought to suffice, the argument goes, since even continuous functions can be computed to arbitrary precision by a Turing machine.

The opposition to this view springs from skepticism about the separation of the mental from the brain-level phenomena. The impulse behind anti-Cartesianism appears to be a reluctance to assign any kind of ontological independence to mind, a reluctance arising from the feeling that mind-talk is but an invitation to all kinds of further mysticisms, such as soul-talk. Thus, the anti-Cartesians tend to be materialists with a vengeance.

Further, in the anti-Cartesian view the brain is nothing like the symbolic processor of the Cartesian. Instead of what is seen as the sequential and combinational perspective of the Symbolic paradigm, some of the theories in this school embrace parallel, "holistic", Non-Symbolic alternatives, while others do not even subscribe to any kind of information processing or representational language in talking about mental phenomena. Those who do accept the need for information processing of some type, nevertheless, reject processing of labeled symbols, and look to analog or continuous processes as the natural medium for modeling the relevant phenomena. In contrast to Cartesian theories, most of the concrete work in these schools deals with perceptual and motor phenomena, but the framework is meant to cover complex cognitive phenomena as well.

Eliminative materialism in philosophy, Gibsonian theories in psychology. and Connectionism in psychology and AI, all can be grouped as more cr less sharing this perspective, even though they differ among each other on a number of issues. The Gibsonian direct perception theory, for example, is non-representational. Perception, in this view, is neither an inference nor a product of any kind of information processing, rather it is a one-step mapping from stimuli to categories of perception, made possible by the inherent properties of the perceptual architecture. All the needed distinctions are already there directly in the architecture, and no processing over representations is needed.

We note that the proponents of the Symbolic paradigm can be happy with the proposition that mental phenomena are implemented by the brain, which may or may not itself have a computationalist account. However, the anti-Cartesian cannot accept this duality. He is out to show the mind as epiphenomenal. To put it simply, the brain is all there is and it isn't a computer either.

Each of these positions that we have described above is really a composite. Few people in either camp subscribe to all the features in our description of them. In particular, many Connectionists may bristle at our inclusion of them on the anti-Cartesian side of the debate. since the descriptions of their work often are in the language of inference and algorithms. We believe that such an algorithmic specification is quite incidental, and does not involve basic representational commitments at the level of discrete symbol systems (see Section 5). In any case, our account helps in understanding the philosophical impulse behind Connectionism, and the rather diverse collection of bedfellows that it has attracted. In fact, Connectionism is a recent and less radical member of of the anti-Cartesian camp. Many Connectionists do not have any commitment to brain-level theory making. It is also explicitly representational — its only argument being the medium of representation.

3.2. Pre- and Quasi-Representational Theories

Let us now trace in a little more detail the various streams in early *AI* that attempted to come to grips with the nature of intelligence. The period under survey can be characterized as a transition from formalisms with an essentially non-representational character through ideas which oscillated between brain-level *vs.* mind-level representations, and finally to a clear dominance of discrete symbolic representations and emphasis on higher cognitive phenomena.

The earliest of the modern attempts in this direction was the Cybernetics stream associated with the work of Wiener (1948) who laid some of the foundations of modern feedback control. The importance of Cybernetics was that it suggested that *teleology could be consistent with mechanism.* The hallmark of intelligence was said to be *adaptation*, and since Cybernetics seemed to provide an answer to how this adaptation could be accounted for with feedback of *information*, and also account for teleology (*e.g.*, "the *purpose* of the governor is to keep the steam engine speed constant"), it was a great source of early excitement for people attempting to model biological information processing. However, Cybernetics never really became the language of AI because it did not have the richness of ontology to talk about cognition and perception. While it had the notion of information processing in some sense, *i.e.* it had goals and mechanisms to achieve them, it lacked the notion of computation not to mention that of representations.

Cybernetics as a *movement* had broader concerns than the issues surrounding feedback control as applied by Wiener to understanding control and communication in animals and machines. Information and automata theories were all part of the Cybernetic milieu of bringing certain biological phenomena under the rigor of formalisms. Modeling the brain as automata (in the sense of automata theory) was another attempt in this tradition to provide a mathematical foundation for intelligence. The finite automata model of nervenets that McCulloch and Pitts (1943) proposed was among the first concrete postulations about the brain as a computational mechanism. These automata models were computational, *i.e.* they had states and state transition functions, and the general theory dealt with what kinds of automata can do what kinds of things. While this was a source of great excitement — one should try to imagine being present at the time when the computer, information theory and the automata theories were all being born at about the same time, and the sense of exhilaration that must have resulted from the thought that a formal language in which to talk about minds and brains was within reach! — in retrospect, automata theory didn't have enough of the right kind of primitive objects for talking about the phenomena of cognition and perception. What AI needed was not theories *about* computation but computational theories of cognition. Naturally enough, automata theory evolved into the formal foundation for some aspects of computer science, but its role in AI *per se* tapered off.

Another strain, which was much more explicit in its commitment to seeking intelligence by modeling its seat, the brain, looked at neurons and neural networks as the units of information processing out of which thought and intelligence can be explained and produced. Neural net simulation and the work on Perceptrons (Rosenblatt, 1962) are two major examples of this class of work. Its lineage can be traced to Hebb's work on cell assemblies which had a strong effect on psychological theorizing. Hebb (1949) proposed a dynamic model of how neural structures could sustain thought, and how simple learning mechanisms at the neural level could be the agents of higher level learning at the level of thought.

In retrospect, there were really two rather distinct kinds of aims that this line of work pursued. In one, an attempt was made to account for the information processing of neurons and neural structures. To the extent that it is generally granted that neural structures form the implementation medium of human intelligence and thought, this seems like an eminently important line of investigation. In fact, over the years, concrete identifications have been made of particular functions computed by particular neural structures in the brain, and these data may eventually form the empirical basis of a theory of how brains and minds can be bridged analytically.

In the other line of work on neural models, prefiguring the claims of modern Connectionism, the attempt was to explain intelligence directly in terms of neural computations. Since in AI explanation of intelligence takes the form of constructing artifacts which are intelligent, this is a rather tall order — the burden of producing programs which simulate neural mechanisms on one hand, and at the same time do what intelligent agents do: perceive, solve problems, explain the world, speak in a natural language, etc., is a heavy one.

Moreover, there is a problem with the level of description — the terms of neural computation seem far removed from the complex content of thought. Bridging this gap without hypothesizing levels of abstraction between neural information processing and highly symbolic forms of thought is difficult. In other words, even if it is true that the brain is made up completely of neural structures of certain types whose behavior is fully understood, and if one is given a bucketful of such neural structures one would still be not very close to constructing a natural language understanding program without theories of knowledge, and syntax and semantics. The general temptation in this area has been to sidestep the difficulties by assuming that appropriate learning mechanisms at the neural level can result in sufficiently complex high level intelligence, much as it presumably occurred in evolution, so that the designer of the artifact need not have theories of cognition or perception at levels higher than the neural level. However, the difficulty of getting the necessary learning to take place in less than evolutionary time has generally resulted in the neural network level not being a serious contender for AI theory formation and system construction until a new generation of Connectionist models began to admit representations of higher level abstractions.

A number of reasons can been cited for the failure of this class of work, *viz.*, Perceptrons and neural nets to hold center stage in AI. The loss of interest in Perceptrons is often attributed to the demonstration by Minsky and Papert (1969) of their inadequacies. However, their demonstration was in fact limited to single layer Perceptron schemes, and was not the real reason for their disappearance from the scene. The real reason, we believe, is that powerful representational and representation manipulation tools were missing.

The alternative of discrete symbolic representations quickly filled this need, and provided an experimental medium of great flexibility. The final transition to Symbolic Computationalism was rather quick. The mathematics of computability also made investigations along this line attractive and productive. The end of the period saw not only a decisive shift towards representational approaches, but the particular kind of representationalism that became the common currency was the Symbolic paradigm.

4. Connectionism and Its Main Features

We turn our attention now to modern Connectionism. While Connectionism as an AI theory comes in many different forms, they all seem share to the idea that the *representation* of information is in the form of *weights* of connections between processing units in a network, and information processing consists of (i) the units transforming their input into some output, which is then (ii) modulated by the weights of connections as inputs to other units. Connectionist theories emphasize a form of learning which is largely in the form of continuous functions adjusting the weights in the network. In some Connectionist theories the above "pure" form is mixed with symbol manipulation processes. Our description is based on the abstraction of Connectionist architectures as described by Smolensky (1988). His description captures the essential aspects of Connectionist framework.

A few additional comments on what constitutes the essential aspects of Connectionism may be useful, especially since Connectionist theories come in so many forms. Our description above is couched in non-algorithmic terms. In fact, many Connectionist theorists describe the units in their systems in terms of algorithms which map their inputs into discrete states. Our view is that the discrete state description of the units' output as well as the algorithmic specification of the units' behavior in a Connectionist network is largely irrelevant (see Section 5). Smolensky's statement that differential equations are the appropriate language to use to describe the behavior of Connectionist networks lends credence to our view. Further, while our description is couched in the form of continuous functions, the essential aspect of the Connectionist architecture is not the property of continuity *per se* (see Section 5), but that the representation medium has no internal labels which are interpreted and no abstract forms which are instantiated during processing.

There are a number of properties of such Connectionist networks that are worthy of note and which explain why Connectionism is viewed as an attractive alternative to the Symbolic paradigm.

- Parallelism: While theories in the Symbolic paradigm are not restricted to serial algorithms Connectionist models are intrinsically parallel, and in most implementations massively parallel.
- Distributedness: Representation of information is *distributed* over the network in a very specialized sense --- the state vector of the weights in the network *is* the representation.
- Softness of constraints (Smolensky, 1988): Because of the continuous space over which the weights take values, the behavior of the network, while not necessarily unimodal, tends to be more or less smooth over the input space.

The two properties of parallelism and distribution have attracted adherents who feel that human memory has a "holistic" character --- much like a hologram --- and consequently have reacted negatively to discrete symbol processing theories, since these compute the needed information from constituent parts and their relations. Dreyfus (1979), for example, has argued that human recognition does not proceed by combining evidence about constituent features of a pattern, but rather uses a "holistic" process. Thus, Dreyfus looks to Connectionism as vindication of his long-standing criticism of Symbolic theories. Connectionism is said to perform "direct" recognition, while Symbolic Computationalism performs recognition by sequentially computing intermediate representations.

The above characteristics are especially attractive to those who believe that AI must be based more on brain-like architectures, even though within the Connectionist camp there is a wide divergence about the degree to which directly modeling the brain is considered appropriate. While some of the theories explicitly attempt to produce neural-level computational structures, some others propose a "subsymbolic level" intermediate between symbolic and neural levels (Smolensky, 1988), and yet others offer connectionism as a computational method that operates in the symbolic level representation itself. The essential idea uniting them all is that the totality of connections defines the information content, rather than representing information as a symbol structure.

5. Is Connectionism Merely An Implementation Theory?

Two kinds of arguments have been made that Connectionism can at best provide possible implementations for Symbolic theories. The traditional one, *viz.*, that Symbolic Computationalism is adequate, takes a couple of forms. In one, continuous functions are thought to be the alternative, and the fact that they can be approximated to an arbitrary degree of approximation is used to argue that one need only consider algorithmic solutions. In the other, Connectionist architectures are thought to be the implementation medium for Symbolic theories, much as the computer hardware is the implementation medium for software. Below we will consider these arguments. We will show that in principle the Symbolic and Non-Symbolic solutions such as Connectionism may be alternative theories in the sense that they may make different representational commitments.

The other argument is based on a consideration of the properties of high level thought, in particular language and problem solving behavior. Connectionism by itself does not have the constructs, the argument runs, for capturing these properties, so at best it can only be a way to implement the higher level functions. We will discuss this and related issues a little later (see Section *6*)

5.1. Symbolic and Non-Symbolic Representations

Let us consider the problem of multiplying two positive integers. We are all familiar with algorithms to perform this task. We also know how the traditional slide rule can be used to do this multiplication. The multiplicands are represented by their logarithms on a linear scale, which are then "added" by being set next to each other, and the result is obtained by reading off the sum's anti-logarithm. While both the algorithmic and slide rule solutions are *representational*, in no sense can either of them be thought of as an "implementation" of the other. They make very different commitments about what is represented. There are also striking differences between them in practical terms. As the size of the multiplicands increases, the algorithmic solution suffers in the amount of *time* it takes to complete the solution, while the slide-rule solution suffers in the amount of *precision* it can deliver.

Let us call the algorithmic and slide-rule solutions C1 and C2. There is yet another solution C3, which is the simulation of C2 by an algorithm. C3 can simulate C2 to any desired accuracy. But C3 has radically different properties from C1 in terms of the information that it represents. C3 is closer to C2 representationally. Its symbol manipulation character is at a lower level of abstraction altogether. Given a blackbox multiplier, ascription of C1 or C2 (among others) as to what is *really* going on makes for different theories about the process. Each theory makes different ontological commitments. Further, while C2 is "analog" or continuous, the existence of C3 implies that the essential characteristic of C2 is not continuity *per se*, but a radically different sense of representation and processing than C1.

An adequate discussion of what makes a symbol in the sense used in computation over symbol systems requires much larger space and time than we have at present, (Pylyshyn (1984) provides a thorough and illuminating discussion of this topic), but the following points seem useful. There is a type-token distinction that seems relevant: symbols are types about which abstract rules of behavior are known and can be brought into play. This leads to symbols being labels which are "interpreted" during the process, while there are no such interpretations in the process of slide rule multiplication (except for input and output). The symbol system can thus represent *abstract forms*, while C2 above performs its addition or multiplication not by instantiating an abstract form, but by having, in some sense, all the additions and multiplications directly in its architecture.

While we have been using the word "process" to describe both C1 and C2, strictly speaking there is no process in the sense of a temporally evolving behavior in C2. The architecture directly produces the solution. This is the intuition behind the Gibsonian direct perception in contrast to the Bruner alternative of perception as inference since the process of inference implies a temporal sequentiality. Whether perception, if it is an inferential process, necessarily has to be continuous with cognitive processes, i.e., they all have access to one knowledge base of an agent is a completely different issue (Fodor, 1983). We mention it here because the perception as inference thesis does not necessarily imply one monolithic process for all the phenomena of intelligence.

Connectionist theories have a temporal evolution, but at each cycle, the information process does not have a step-by-step character like algorithms do. Thus, the alternatives in the non-symbolic paradigm are generally presented as "holistic." The Connectionist models stand in the same relationship to the symbolic models that C2 does to C1. The main point is that there exists functions for which Symbolic and Non-Symbolic accounts differ fundamentally in terms of representational commitments. Having granted that Connectionism (actually, Non-Symbolic theories in general) can make a theoretical difference, we now want to argue that the difference Connectionism makes is relatively small to the practice of most of AI.

6. Information Processing Abstractions

6.1. Need for Compositionality

Proponents of Connectionism often claim that solutions in the Symbolic paradigm are composed from constituents, while Connectionist solutions are "holistic", *i.e.* they cannot be explained as compositions of parts. Composition. in this argument, is taken to be intrinsically a Symbolic Computational process. Certainly, for some simple problems there exist Connectionist solutions with this "holistic" character. There are Connectionist solutions to character recognition, for example, which directly map from pixels to characters and which cannot be explained as composing evidence about the features such as closed curves, lines and their relations. Character recognition by template matching, though not a Connectionist solution, is another example whose information processing cannot be explained as feature composition. However. as problems get more complex, the advantages of modularization and composition are as important for Connectionist approaches as they ar· 'or Symbolic Computation or for Civil Engineering for that matter.

A key point is composition may be done Connectionistically, *i.e.* it does not always require S[illegible]abolic Computational methods. To see this, let us consider word recognition, a problem area which has attracted significant attention in Connectionist literature. Let us consider recognition of the word "TAKE" as discussed by McClelland, Rumelhart and Hinton (1986). A "featureless" Connectionist solution similar to the one for individual characters can be imagined, but a more natural one would be one which in some sense composes the evidence about individual characters into a recognition of the word. In fact, the Connectionist solution in that McClelland, Rumelhart and Hinton describe has a natural interpretation in these terms. The fact that the word recognition is done by composition does not mean either that each of the characters is explicitly recognized as part of the procedure, or the· the evidence is added together in a step by step, temporal sequence.

Why is such a [illegible]itional solution more natural? Reusability of parts, reduction in learning co[illegible]ty as well as greater robustness due to intermediate evidence are the major c[illegible]tional advantages of modularization. If the reader doesn't see the power of n[illegible]larization for word recognition, he she can consider sentence recognition and see [illegible]at if one were to go directly from pixels to sentences, without in some sense going through words, the number of recognizers and their complexity would have to be very large even for sentences of bounded length. To put it differently, if one has a system that already recognizes "Monkey," "banana," and "Eat(a, b)", then recognizing "Monkey eats banana." without composing the constituent recognizing capabilities above would be very wasteful of resources and would require excessive learning times as well. Composition is a powerful aid against complexity whether the underlying system is Connectionist or Symbolic. Of course, Connectionism provides one style for composition and Symbolic methods another, each with its own "signature" in terms of the details of performance.

These examples also raise questions about the claims of distributedness of Connectionist representations. For complex tasks, information is in fact localized into portions of the network. Again, in the network for recognition of the word "TAKE" physically local subnets can be identified, each corresponding to one of the characters. Thus, the hopes of some proponents for almost holographic distributedness of representation are bound to be unrealistic.

6.2. The Information Processing Level

Marr (1982) originated the method of information processing (IP) analysis as a way of separating the essential elements of a theory from implementation level commitments. He proposed that the following methodology be adopted for this purpose. First, identify an IP function with a clear specification about what kind of information is available for the function as input and what kind of information needs to be made available as output. Then, specify a particular IP theory for achieving this function by stating what kinds of information need to be represented at various stages in the processing. Actual algorithms can then be proposed to carry out the IP theory. These algorithms will make additional representational commitments. In the case of vision, for example, Marr specified that one of the functions is to take as input image intensities in a retinal image, and produce as output a 3-dimensional shape description of the objects in the scene. His theory of how this function is achieved in the visual system is that three distinct kinds of information need to be generated: from the image intensities, a primal sketch of significant intensity changes -- a kind of edge description of the scene -- is generated, then a description of surfaces of the objects and their orientation, what he called a 2 1/2 -dimensional sketch is produced from the primal sketch, and finally a 3-dimensional shape description is generated. Even though Marr talked in the language of algorithms as the way to realize the IP theory, there is in principle no reason why portions of the implementation cannot be done Connectionistically.

Information processing level abstractions constitute the top level content of much AI theory formation. In the example about recognition of the word "TAKE" in the previous section, the IP level abstractions in terms of which the theory of word recognition was couched were the evidences about the presence of individual characters. The difference between schemes in the Symbolic and Connectionist paradigms is that these evidences are labeled symbols in the former, which permit abstract rules of compositions to be invoked and instantiated, while in the latter they are represented more directly and affect the processing without undergoing any interpretive process. Interpretation of a piece of a network as evidence about a character is a design and explanatory stance, and is not part of the actual information processing.

We claim that as Connectionist structures are built to handle increasingly complex phenomena, they will end up having to incorporate their own versions of *modularity* and *composition*. Already we saw this in the only moderately complex word recognition example. When, and if, we finally have Connectionist implementations solving a variety of high level cognitive problems (say natural language understanding or problem solving and planning), the design of such systems will have an enormous amount in common with the corresponding Symbolic theories. This commonness will be at the level of information processing abstractions that both classes of theories would need to embody. In fact, the *content* contributions of many of the nominally Symbolic theories in AI are really at the level of the IP abstractions to which they make a commitment, and not to the fact that they were implemented in a symbolic structure. Symbols have often merely stood in for *abstractions* that need to be captur ' one way or another, and have often been used as such. The hard work of theory making in AI will always remain at the level of proposing the right IP level of abstractions, since they provide the content of the representations. The decisions about which of the IP transformations are best done by means of connectionist networks, and which using symbolic algorithms, can properly follow once the IP level specification of the theory has been given. Thus, the Connectionist and the Symbolic approaches are both *realizations* of a more abstract level of description, *viz.*, the *information processing level.*

6.3. Learning to the Rescue?

What if Connectionism can provide learning mechanisms such that one starts without any IP abstractions represented, and the system learns to perform the task in a reasonable amount of time? In that case, Connectionism can sidestep pretty muc ıll the representational problems and dismiss them as the bane of Symbolic Com: ɾationalism. The fundamental problem of complex learning is the *credit assignment problem*, *i.e.*, the problem of deciding what part of the system is responsible for either the correct or the incorrect performance in a case, so that the learner knows how to change the structure of the system. Abstractly, the range of variation of the structure of a system can be represented as a multi-dimensional space of parameters, and the process of learning as a search process in that space for a region that corresponds to the right structure of the systems. The more complex the system, the vaster the space in which to do the search. Thus, learning the correct set of parameters by search methods which do not have a powerful notion of credit assignment would work in small search spaces, but would be computationally prohibitive for realistic problems. Does Connectionism have a solution to this problem?

If one looks at particular Connectionist schemes that have been proposed for some tasks such as learning tense endings (Rumelhart and McClelland, 1986b), a significant part of the abstractions needed are built into the architecture in the choice of inputs, feedback directions, allocation of subnetworks, and the semantics that underlie the choice of layers for the Connectionist schemes. Thus, the inputs and the initial configuration incorporate a sufficiently large part of the abstractions needed that what is left to be discovered by the learning algorithms, while non-trivial, is proportionately small. The initial configuration decomposes the search space for learning in such a way that the search problem is much smaller in size. In fact, the space is sufficiently small that statistical associations can do the task.

The recognition scheme for "TAKE" again provides a good example for illustrating this point. In the Connectionist scheme that we cited earlier the decisions about which subnet is going to be largely responsible for "T", which for "A", etc., as well as how the feedback is going to be directed are all essentially made by the experimenter before any learning starts. The underlying IP theory is that evidence about individual characters is going to be formed directly from the pixel level, but recognition of "TA" will be done by combining information about the presence of "T" and "A", as well as their joint likelihood. The degree to which the evidence about them will be combined is determined by the learning algorithm and the examples. In setting up the initial configuration, the designer is actually programming the architecture to reflect the above IP theory of recognizing the word. An alternate theory for word recognition, say one that is more "holistic" than the above theory, *i.e.* one that learns the entire word directly from the pixels, will have a different initial configuration. Of course, because of lack of guidance from the architecture about localizing search during learning, such a network will take a much longer time to learn the word. That precisely is the point: the designer recognized this and set up the configuration so that learning can occur in a reasonable time. Thus, while the Connectionist scheme for word recognition still makes the useful *performance* point about Connectionist architectures for problems that have been assumed to require a Symbolic Computational implementation, a significant part of the leverage still comes from the IP abstractions that the designer started out with, or have been made possible by an earlier learning phase working with highly structured configurations.

Additionally, the system that results after learning has a natural interpretation in terms of the abstractions that are needed to solve the problem: the learning process can be interpreted as having successfully searched the space for those additional abstractions that are needed to solve the problem. Thus, Connectionism is one way to map from one set of abstractions to a more structured set of abstractions. Most of the representational issues remain, whether or not one adopts Connectionism for such mappings.

Of course in human learning, while some of the abstractions needed are "programmed" in at various times through explicit instruction, a large amount of learning takes place without any "designer" intervention in setting up the learning structure as we described in the "TAKE" example. However, there is no reason to believe that humans start with a structure- and abstraction-free initial configuration. In fact, in order to account for the power of human learning, the initial configurations that a child starts out with will need to contain complex and intricate representations sufficient to support the learning process in a computationally efficient way.

7. The Domains for Connectionism and Symbolic Computationalism

7.1. Macro- and Micro- Phenomena

Rumelhart and McClelland (1986a), use the term "micro-" in the subtitle of their book to indicate that the Connectionist theories that they are concerned with deal with the fine details of intelligent processes. A duration of 50-100 milliseconds has often been suggested as the size of the temporal "grain" for processes at the micro level. Macro-phenomena take place over seconds if not minutes in the case of a human. These evolve over time in such a way that there is a clear temporal ordering of some of the major behavioral states. As an example, let us consider the problem solving behavior of GPS (Newell and Simon, 1972). The agent is seen to have a goal at a certain instant, to set up a subgoal at another instant, and so on. Within this problem solving behavior, the selection of an appropriate operator, which is typically modeled in GPS implementations as a retrieval algorithm from a Table of Connection, could be a "micro" behavior. Many of the phenomena of language and reasoning have a large macro component. Thus, the domain of macro-phenomena includes, but is not restricted to, phenomena whose markings are left in consciousness as a temporal evolution of beliefs, hypotheses, goals, subgoals, etc. Neither traditional Symbolic Computationalism nor radical Connectionism has much use for this distinction since all the phenomena of intelligence, micro and macro, are meant to come under their particular purview.

We would like to present an alternative case for a division of responsibility between Connectionism and Symbolic Computationalism in accounting for the phenomena of intelligence. The architectures in the Connectionist mold offer some elementary functions which are rather different from those assumed in the traditional Symbolic paradigm. By the same token, the body of macro phenomena seems to us to have a large symbolic and algorithmic content. A proper integration of these two modes of information processing can be a source of powerful explanations of the total range of the phenomena of intelligence.

We are assuming it as a given that much of high level thought has a symbolic content to it (see (Pylyshyn, 1984) for arguments that make this conclusion inescapable). *How much* of language and other aspects of thought require this can be matter of debate, but certainly logical reasoning should provide at least one example of such behavior. We are aware that a number of philosophical hurdles stand in the way of asserting the symbolic content of conscious thought. If one is a radical behaviorism or a non-representationalist, we see that no advantage accrues from granting that the corpus of thought, including language and logical reasoning, has a symbolic structure. Saying that all that passes between people when they converse is airpressure exchanges on the eardrum has its charms, but we will forego them in this discussion.

Asserting the symbolic content of macro phenomena is not the same as asserting that the internal language and representation of the processor that generates them has to be in the same formal system as that of its external behavior. The traditional Symbolic paradigm has made this assumption as a working hypothesis.

which Connectionism challenges. Even if this challenge is granted there is still the problem of figuring out how to get the macro behavior out of the Connectionist structure.

7.2. Symbolic Theories as Approximations?

Rumelhart and McClelland (1986a) comment that Symbolic theories that are common in AI are really explanatory approximations of a theory which is Connectionist at a deeper level. Let us consider the "TAKE" example again. Saying that the word is recognized by combining evidences about individual characters in a certain way may appear to be giving an Symbolic Computational account, but this description is really neutral regarding whether the combination is to be done Connectionistically or Symbolic Computationally. It is not that Connectionist structures are the reality and Symbolic accounts provide an explanation, it is that the IP abstractions contain a large portion of the explanatory power.

As another example of this let us consider the suggestion by Rumelhart, Smolensky, McClelland and Hinton (1986) that a schema or a frame is not really explicitly represented as such, but is constructed as needed from more general Connectionist representations. We are in complete agreement with this view. However, this does not mean to us that schema theory is only a macro approximation. Schema, in the sense of being IP abstractions needed for certain macro phenomena, is a legitimate conceptual construct, for encoding of which Connectionist architectures offer a particularly interesting way.

7.3. Conscious and Intuitive Processors

Fodor and Pylyshyn (1987) have argued that much of thought has the properties of *productivity* and *systematicity*. Productivity refers to a potentially unbounded recursive combination of thought that is possible in human intelligence. Systematicity refers to the capability of combining thoughts in ways that require abstract representation of underlying forms. Connectionism may provide some of the architectural primitives for performing parts of what is needed to achieve these characteristics, but cannot be an adequate account in its own terms. We need computations over symbol systems, their capacity for abstract forms and algorithms, to realize these properties.

In order to account for the highly symbolic content of conscious thought and to place Connectionism in a proper relation to it, Smolensky (1988) proposes that Connectionism operates a lower level than the symbolic, a level he calls *subsymbolic*. He also posits the existence of a *conscious processor* and an *intuitive processor*. The Connectionist proposals are meant to apply directly to the latter. The conscious processor may have algorithmic properties, according to Smolensky, but still a very large part of the information processing activities that have been traditionally attributed to Symbolic architectures really belong in the intuitive processor.

A complete Connectionist account in our view needs to account for how a sub- or non- :mbolic structure integrates smoothly with a higher level process that is heavily s\ )olic. There is the additional problem that an integrated theory has to face. E\ :: if thought were merely epiphenomenal, we know that the phenomena of conscious ·ess have a causal interaction with the behavior of the intuitive processor. What we consciously learn and discuss and think affects our unconscious behavior slowly but surely, and vice versa. What is conscious and willful today becomes unconscious tomorrow. All this raises a more complex constraint for Connectionism: it now needs to provide some sort of continuity of representation and process so that this interaction can take place smoothly.

7.4. Architecture-Independent and -Dependent Decompositions

We argued, in Section *5*, that given a function, the approaches in the Symbolic and Non-Symbolic paradigms may make rather different representational commitments; in compositional terms, they may be composing rather different subfunctions. In Section *6* we argued, seemingly paradoxically, that for complex functions the two theories converge in their representational commitments. A way to clarify this is to think of two stages in the decomposition: an architecture-independent and an architecture-dependent one. The former is an IP theory that will be realized by particular architectures for which additional decompositions will need to be made. Simple functions such as multiplication (of Section *5*) are so close to the architecture level that we only saw the differences between the representational commitments of the algorithmic and slide rule solutions. The word recognition problem (of Section *6*) is sufficiently removed from the architectural level that we saw macro-similarities between Symbolic Computationalist and Connectionist solutions. The final performance will of course have micro-features that are characteristic of the architect::r such as the "softness of constraints" for Connectionist architectures.

Where the archi. ·e-independent theory stops and the architecture-dependent starts does not ιve a clear line of demarcation. It is an empirical issue, partly related to the primitive functions that can be computed in a particular architecture. The farther away a problem is from the architectures' primitive functions, the more ar: ·ecture-independent decomposition needs to be done at design time.

Connectionist and Symbolic Computationalist functions, in our view, have different but overlapping domains. The basic functions that the Connectionist architecture delivers are of a very different kind than have been assumed so far in Symbolic paradigm, and IP theories need to take this into account in their formulations. A number of investigators in AI who work at the IP level correctly feel the attraction of Connectionist theories for some parts of their theory formation. The impact of Connectionism is being felt in identifying some of the component processes of IP theories as places where a Connectionist account seems to accord better with intuitions. We believe that certain kinds of retrieval and matching operations. and low level parameter learning by searching in local regions of space are especially appropriate tasks for which the higher level IP theories may choose

Connectionist alternatives if the fine points of performance are of theoretical importance. However, even here one should be careful about putting too much faith in Connectionist mechanisms *per se*. As we have stated earlier, the power for even these operations is going to come from appropriate encodings that get represented Connectionistically. Thus, while memory retrieval may have interesting Connectionist components to it, the basic problem will still remain the principles by which episodes are indexed and stored, except that now one might be open to these encodings being represented Connectionistically.

8. Conclusions

With regard to general AI and Connectionism's relevance to it, we would like to say, as H. L. Mencken is alleged to have said in a different context, "There is something to what you say, but not much." Much of AI research, except where microphenomena dominate and Symbolic AI is simply too hard-edged in its performance, will and should remain largely unaffected by Connectionism. We have given two reasons for this. One reason is that most of the work is in coming up with the information processing theory of a phenomenon in the first place. The more complex the task is the more common are the representational issues between Connectionism and the Symbolic paradigm. The second reason is that none of the Connectionist arguments or empirical results show that the symbolic, algorithmic character of thought is either a mistaken hypothesis, purely epiphenomenal or simply irrelevant.

Our arguments for and against Connectionist notions are not really specific to Connectionist architectures that have been proposed. The arguments apply generally to other Non-Symbolic approaches as well, *e.g.* all sorts of analog computers. Connectionist architectures, especially those that deny modeling the brain level, often seem to have an air of arbitrariness about them, since it is then not clear what the constraints are: why that rather than something else? However, in fairness, these architectures ought to be viewed as exploratory, and in that sense they are contributing to our understanding of the capabilities and limitations of alternatives to the symbolic paradigm.

It seems to us that we need to find a way to accept three significant insights about mental architectures:

- (i) A large part of the relevant content theory in AI has to do with the *what* of mental representations. We have called them the information processing abstractions.

- (ii) Whatever one's position on the nature of representations below conscious processes, it is clear that processes at or close to that level are intimately connected to language and knowledge, and thus have a large discrete symbolic content.

- (iii) The Connectionist ideas on representation suggest how non- symbolic representations and processes may provide the medium in which thought resides.

Acknowledgments

We gratefully acknowledge the support of the Defense Research Projects Agency, RADC Contract F30602-85-C-0010, and the Air Force Office of Scientific Research, grant 87-0090, during the preparation of this paper. In addition, one of the authors (Allemang) would like to acknowledge the support of a National Science Foundation fellowship during this period. This paper has benefited from discussions with Tom Bylander, Jerry Fodor, John and Susan Josephson, Hari Narayanan, Peter Sandon, and Michael Weintraub. We would like to thank them all.

References

Bruner, J. S. (1957). On Perceptual Readiness. Psychological Review, 64: 123-152.

Dennett, D. (1986). The Logical Geography of Computational Approaches: A View from the East Pole. In Brand and & Harnish (eds.) *The Representation of Knowledge and Belief*, Tucson, AZ: The University of Arizona Press.

Dreyfus, H. L. (1979). *What Computers Can't Do*, 2nd Edition. New York: Harper & Row.

Fodor, J. A. (1983). *The Modularity of Mind*. Cambridge, MA: The MIT Press, A Bradford Book.

Fodor, J. A. & Pylyshyn, Z. W. (1987). Connectionism and Cognitive Architecture: A Critical Analysis. (draft, to appear).

Hebb, D. O. (1949). *The Organization of Behavior*. New York: Wiley.

Marr, D. (1982). *Vision*. San Francisco: Freeman.

McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. (1986). The Appeal of Parallel Distributed Processing. In Rumelhart, McClelland and the PDP Research Group (eds.), *Parallel Distributed Processing, Volume 1*, Cambridge, MA: M.I.T. Press, A Bradford Book.

McCulloch, W., and Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics. 5: 115-137.

Minsky, M. L. and Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Newell, A. (1980). Physical symbol systems. Cognitive Science. 4: 135-183.

Newell, A, and Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, New Jersey: Prenctice-Hall.

Pylyshyn, Z. W. (1984). *Computation and Cognition: Towards a Foundation for Cognitive Science*. Cambridge, MA: MIT Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Cornell Aeronautical Laboratory Report 1196-G-8. Washington, D.C.: Spartan.

Rumelhart, D. E. & McClelland, J. L. (1986a). PDP Models and General Issues in Cognitive Science. In Rumelhart, McClelland and the PDP Research Group (eds.). *Parallel Distributed Processing, Volume 1*, Cambridge, MA: M.I.T. Press, A Bradford Book.

Rumelhart, D.E. & McClelland, J.L. (1986b). On Learning the Past Tenses of English Verbs. In McClelland, Rumelhart and the PDP Research Group (eds.). *Parallel Distributed Processing, Volume 2*, Cambridge, MA: M.I.T. Press, A Bradford Book.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., and Hinton, G. E. (1986). Schemata and Sequential Thought Processes in PDP models. In McClelland, Rumelhart and the PDP Research Group (eds.). *Parallel Distributed Processing, Volume 2*, Cambridge, MA: M.I.T. Press, A Bradford Book.

Smolensky, P. (1988). On the Proper Treatment of Connectionism. *The Behavioral and Brain Sciences, 11*, (in press).

Wiener, N. (1948). *Cybernetics, or Control and Communication in the Animal and the Machine*. New York: Wiley.

Extended Abstract

On Some Experiments with Neural Nets

JOHN KOLEN
ASHOK GOEL
DEAN ALLEMANG

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

February, 1988

[illegible]

On Some Experiments with Neural Nets

JOHN KOLEN, ASHOK GOEL, and DEAN ALLEMANG

Department of Computer and Information Science
The Ohio State University

Extended Abstract

A variety of neural networks that purport to model aspects of motor, perceptual, and language phenomena have been recently reported in the Artificial Intelligence literature. However, despite the modest success of these models, it is not yet entirely clear where the computational power of neural networks comes from. Let us consider the specific case of neural networks in the connectionist mold [4]. It has been claimed that the computational power of connectionist networks emerges from representing knowledge as numerical weights of connections between the processing units. However, it has been argued [1] that while the medium of representation in connectionist networks is indeed different, the real computational power lies in the *information processing abstractions* that form the content of *representation*. It has been claimed that the power of connectionist networks comes from the use of "hidden" units. However, it has been argued [2] that the real role of the hidden units is to capture the needed abstractions. It has been claimed that the power lies in the learning mechanisms such as the generalized delta rule, and back propagation of corrective feedback. However, it has been shown [3] that the generalized delta rule is only a more general form of the well known hill climbing procedure, and back propagation is merely a recursive application of this procedure.

In an effort to identify precisely where the computational power in connectionist networks comes from, we have conducted a small set of experiments. Our strategy has been to consider simple information processing tasks, and study them systematically and exhaustively. One of the tasks that we have studied is computation of the exclusive-OR Boolean function. Rumelhart and McClelland [5] have reported on a connectionist network for this task. Their network learned to compute exclusive-OR in a few hundred training sessions. However, when we repeated their experiment with an initially random set of weights it took the same network a few hundred thousand training sessions before it learned to compute exclusive-OR correctly. We tried a variety of learning rules and back propagation techniques but with little success, until, by chance, we hit upon the just the right set of initial weights when the network did indeed converge to the correct set of weights in only a few hundred training sessions.

We have conducted a similar experiment with a connectionist network that learns to play *tic-tac-toe*. Rumelhart *et al.* [6] have reported on such a network. Their network learns to play the game perfectly in a few hundred training sessions. However, the needed abstractions (*row*, *column*, etc.) are *explicitly* represented in their network, which begs the question that we are asking. Instead, we designed a connectionist network similar to theirs but without any hard-wired abstractions. When we repeated their experiment with an initially random set of weights, treating the number of hidden units as a parameter of the network, our network showed little learning. Again we tried a variety of learning mechanisms and back propagation techniques but with little success, until, again by accident, we hit upon the just the right combination of hidden units and initial weights when it took our network a few hundred thousand training sessions before it learned to play *tic-tac-toe* well, and even then its performance was imperfect.

What these experiments demonstrate is that the computational power of neural networks lies not so much in the representation medium, or hidden units, or learning mechanisms --- although they do make a difference. Instead, in order to make a network perform a given task computationally efficiently one of two things has to be done. Either the needed information processing abstractions have to be represented explicitly in the network as Rumelhart *et al.* do with their network for playing *tic-tac-toe*. Alternatively, the needed abstractions have to be captured implicitly by selecting the right number of hidden units and the right set of initial weights as Rumelhart *et al* do with their network for computing the exclusive-OR function. It is these abstractions that reduce the size of learning space and guide the network in the navigation of this space.

Acknowledgments

The research reported in this paper has benefited from discussions with B. Chandrasekaran to whom we are deeply grateful. We are grateful also for the support of the Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010, and the Air Force Office of Scientific Research, grant 87-0090.

References

[1] B. Chandrasekaran, Ashok Goel and Dean Allemang. "Connectionism and Information Processing Abstractions. To appear in *Proceedings of the AAAI Symposium on Parallel Models of Intelligence*, Stanford University, March 1988. A shorter version is to appear as a commentary in the *Journal of Brain and Behavioral Sciences*. United Kingdom: Cambridge University Press, 1988.

[2] B. Chandrasekaran and Ashok Goel. "From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task". To appear in *IEEE Transactions on Systems, Man, and Cybernetics*, 1988.

[3] Marvin Minsky and Seymour Papert. *Epilogue* in *Perceptrons: An Introduction to Computational Geometry; Expanded Edition*. Cambridge, MA: MIT Press, 1988.

[4] David Rumelhart, James McClelland, and the PDP Research Group (editors). *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volumes 1 and 2*. Cambridge, MA: MIT Press, A Bradford book, 1986.

[5] David Rumelhart, Geoffrey Hinton, and Ronald Williams. "Learning Internal Representation by Error Propagation". In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 1*, David Rumelhart, James McClelland, and the PDP Research Group (editors). Cambridge, MA: MIT Press. A Bradford book. 1986.

[6] David Rumelhart, Paul Smolensky, James McClelland, and Geoffrey Hinton. "Schemata and Sequential Thought Processes in PDP Models". In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 2*, David Rumelhart, James McClelland, and the PDP Research Group (editors). Cambridge, MA. MIT Press. A Bradford book. 1986.

88-BC FRMNOM

The Ohio State University
Department of Computer and Information Science
Laboratory for Artificial Intelligence Research

Technical Report
January 1988

FROM NUMBERS TO SYMBOLS TO KNOWLEDGE STRUCTURES: ARTIFICIAL INTELLIGENCE PERSPECTIVES ON THE CLASSIFICATION TASK

B. Chandrasekaran and Ashok Goel
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

Note: This paper is to appear in IEEE Transactions on Systems, Man and Cybernetics, 1988.

From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task

B. CHANDRASEKARAN and ASHOK GOEL

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

January, 1988

This paper is to appear in IEEE Transactions on Systems, Man and Cybernetics, 1988. An earlier version of the paper appears in *Pattern Recognition in Practice II*, E.S. Galsema and L.N. Kanal: editors, pages 547-559, North-Holland, 1985.

From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task

B. CHANDRASEKARAN and ASHOK GOEL

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

Abstract

We consider the very general information processing task of *classification*, and review it from the perspectives of the knowledge-based reasoning, pattern recognition, and connectionist paradigms in Artificial Intelligence, paying special attention to knowledge-based classificatory problem solving. We trace the evolution of the mechanisms for classification as the computational complexity of the problem increases, from numerical parameter setting schemes, through those using intermediate abstractions and then relations between symbols, and finally to complex symbolic structures which explicitly incorporate domain knowledge. The paper can be viewed as a bridge-building activity, describing the approaches of three different research communities to the same general task.

I. INTRODUCTION

Classification is a very general information processing task in which specific entities are mapped onto general categories. As the amount of data about the entity to be classified and the number of classificatory categories increase, typically so does the computational complexity of the task. In this paper, we review the classification task from the perspectives of the knowledge-based reasoning, pattern recognition, and connectionist paradigms in Artificial Intelligence (AI), paying special attention to knowledge-based classificatory problem solving. We trace the evolution of the mechanisms for classification as the complexity of the problem increases, from numerical parameter setting schemes, through those using intermediate abstractions and then relations between symbols, and finally to complex symbolic structures which explicitly incorporate domain knowledge. The paper can be viewed as a bridge-building activity, describing the approaches of three different research communities to the same general task. It can also be viewed as an attempt, by using the classification task as a concrete example, to give an intuitive account of how the information processing activity underlying thought necessarily evolved into complex symbolic processes in order to handle increasing complexity of problems and requirements of flexibility.

II. THE CLASSIFICATION TASK

Classification, sometimes called categorization in the cognitive science literature, as an information processing task can be functionally specified by the information it takes as input, and the information it gives as output. In its general form, the input to the classification task is a collection of data about some specific entity (*e.g.*, an object, a state, a case, or a situation), and the output is the general category (or categories) pertaining to the entity. We note that this characterization of the classification *task* as a map from specific entities to general categories makes no commitments to the *mechanism* by which the mapping is to be accomplished. Classification has been an active research issue in the knowledge-based reasoning, pattern recognition, and connectionist paradigms, though the paradigms differ in the mechanisms by which the task is performed.

A. Classification and Knowledge-Based Systems

The area of knowledge-based reasoning, though of relatively recent origin, is already a well established paradigm in AI. The essential idea of the field is to capture in computer programs, explicitly and in symbolic form, the *knowledge and problem solving methods* of human experts for selected domains and tasks. In fact, because of the central role of explicit domain knowledge of human experts, the field is often called expert systems. This is not an appropriate place to discuss the general issues of knowledge representation and problem solving in the area of knowledge-based systems, many of which remain open and active research issues. There are many expert tasks that have been successfully emulated by these systems; there are an even larger number of things that human experts do that are beyond the current state of technology for construction of knowledge-based systems. Nevertheless, when we examine the intrinsic nature of the *tasks* that knowledge-based systems perform, a surprising fact emerges: many of them solve variants of problems which are intrinsically *classificatory* in nature. We are not suggesting here that the authors of these programs *recognized* them as classification problems and used methods appropriate to the classification task, but that independent of how they were solved the problems have an intrinsically classificatory character. Let us consider some examples:

- The MYCIN system [35], in its diagnostic phase, has the task of classifying patient data onto an infectious agent hierarchy, *i.e.*, the diagnostic task is identification of an infectious agent category, as specific as possible, that pertains to the patient data.
- The PROSPECTOR system [14] classifies a geological description as corresponding to one or more mineral formation classes.
- The SACON System [3] classifies structural analysis problems into categories for each of which a particular family of analytical methods is appropriate.
- The MDX system [6], [8], [20] *explicitly* views a significant portion of the diagnostic task as classifying a complex symbolic description (the patient data) as an element, as specific as possible, in a disease classification hierarchy.

We do not mean to imply that all problems are classification problems, or that they can be usefully converted into such problems. R1 [27] and AIR-CYL [5], *e.g.*, perform different versions of the *object synthesis* problem, *i.e.*, simple versions of the design problem. Dendral [4], Internist [30] and RED [22] are different systems all performing various versions of *abductive assembly of composite explanatory hypotheses*. Chandrasekaran [7], [9], [10], has provided taxonomies of such *generic tasks*, and has identified classification as one of them. Recently, Clancey [12] has made a similar assessment of how several knowledge-based systems perform classificatory problem solving.

B. Classification and Pattern Recognition Models

The area of pattern recognition, now nearly thirty years old, represents another paradigm in AI. The classification task has been intimately associated with pattern recognition models from the very beginning of the field. In fact, in the early days of AI, the problem of recognition was *formulated* as a problem of classification, in particular one of statistical classification of pattern vectors onto one of a finite number of categories, each category characterized by some kind of probability distribution. Indeed, what started out as a practically useful formulation became so dominant that there was a need for a paper such as that by Kanal and Chandrasekaran [23] pointing out that classification is only one of the formulations for the more general recognition problem. Even when newer techniques such as syntactic techniques came into the field, the problem was still often formulated as a classification problem, this time into grammatical categories.

C. Classification and Connectionist Networks

"Neural" modeling, which predates the early perceptron models and appears to be undergoing a revival in its modern "connectionist" version, is still another paradigm in AI. The essential idea in this area is to represent knowledge as numerical weights of connections between units in a network. A variety of neural models, from linear threshold, digital networks [15], [32], to non-linear analogue architectures [21], have been developed. These models typically deal with motor or perceptual phenomena; neural networks that capture a range of complex, higher-level cognitive processes have yet to be proposed. Although our remarks are intended to be more generally applicable, in this paper we will confine our discussion only to linear threshold, digital networks in the connectionist mold in which the emphasis is on the memory and learning aspects of reasoning.

The earlier connectionist networks, *e.g.*, the perceptron model, were once viewed as devices for practical visual pattern recognition, and since the problem of pattern recognition itself was viewed as that of classification, perceptrons were really classificatory devices. The important role of classification is evident even in the more recent connectionist architectures, in which "hidden" units separate the input and the output units. Let us consider, as an example, the MBRtalk system [37], a connectionist scheme for the task of word pronunciation. It uses a numerical relaxation technique for problem solving, and a method for back propagation of corrective feedback during learning. The important point for our purposes, however, is that MBRtalk performs its task by *classifying* character substrings of the input words onto phonemes.

III. The Ubiquity of Classification

There are two things that are important to note from the above discussion: firstly, classification appears to be a rather ubiquitous information processing task, and secondly, classification has been an important research issue in the various paradigms in AI. This suggests that classification is not an artifact of any one point of view, but rather a "natural kind" of information processing task of considerable cognitive significance. Indeed, classification appears to be a powerful human strategy for organizing knowledge for comprehension and action. The human tendency to classify input entities is so strong that we often classify without necessarily being consciously aware of it, and feel we have accomplished something by merely naming entities as categories, even if we cannot do much about it. The use of classification as a strategy for knowledge organization can be found in virtually every area of human intellectual activity. In Biology, *e.g.*, taxonomic classification has long been an important methodology for organization of knowledge, and recently, mathematical techniques has been pressed into service for providing better classification in this field [36]. Some of the more recent controversies regarding evolutionary biology, *e.g.*, the traditional gradual evolutionary *vs.* the punctuated equilibrium theories, also revolve around implications of various theories of biological classification. The periodic table of chemical elements is another common classification structure in which first groups of elements and then the specific elements are identified.

A. The Computational Power of Classification

A simple computational explanation can be given for the importance of classification as an *information processing strategy*. We can think of a general task of an intelligent agent as performing actions on the world for achieving certain goals, where the right action for accomplishing a specific goal typically is a function of the relevant states of the world. In the medical domain, for example, we may view the general problem facing the physician as that of finding an appropriate therapeutic action for a given set of symptoms that describes the state of a patient and is a subset of the set of all possible symptoms. One way of mapping states of the world to actions on it might be to use a *decision table* that relates various subsets of state variables to the action variable. However, if there are n state variables $v_1, v_2, \ldots, v_n$, each of which may take on one of q values, then both the time and space complexities of mapping the states onto actions by table look-up are *$O(n.q^n)$* [17]. Thus, the table look-up approach to making decisions about actions on the world would be useful only for very small problems. In fact, the cardinality of the relevant states of the world generally is very large, *e.g.*, in the medical domain, the total number of possible states of a patient is the cartesian product of the distinct values for each of the state variables (symptoms, values from laboratory tests, other manifestations etc.). Thus, for complex, real world problems such as medical problem solving the decision table is bound to be too large for construction, storage, looking up, and modification.

The general problem of finding the right action may be solved more efficiently, however, if action knowledge can be *indexed*, not by the states of the world, but *by equivalence classes of states of the world*. A physician's therapeutic knowledge, *e.g.*, may be indexed not directly by the detailed values of the patient state variables, but by diseases, each of which can be thought of as defining an equivalence class of patient state variables. What we are suggesting here is that a *functional decomposition* of mapping states of the world to actions on it into first mapping the states onto their equivalence classes, and then using these classes for indexing the right actions often results in substantial reduction in the computational complexity of the problem since the number of equivalence classes typically is much smaller than the total number of states. The classification task corresponds to the first component in this decomposition, in which specific entities such as states of the world are mapped onto general categories which represent their equivalence classes. Medical problem solving thus may be organized first as classifying patient symptoms onto disease categories, *i.e.*, diagnosis as classification, and then indexing the therapeutic actions by the disease categories. It may not, of course, always be possible to decompose the general problem of finding the right action in such a manner; however, whenever possible, it is computationally advantageous to do so. The decomposition of mapping states of the world to actions on it is illustrated by the JESSE system [18], which supports a simple version of political decision making. JESSE first classifies the state variables describing a given situation onto situation assessment categories, and then uses these categories to index appropriate policies for action from a store of policy options.

B. Classificatory Categories

Classificatory categories represent the equivalence classes of entities that are input to the classification task. Much of human thinking is organized around classification, both in terms of acquiring new classificatory categories, and using existing categories to perform classifications, since classification provides a substantial computational advantage in solving problems. In knowledge-based systems, the classificatory categories typically are labeled symbolically, and often correspond to *concepts* in the task domain. In connectionist networks on the other hand, no labels are associated with the categories, and the categories do not necessarily correspond directly with the domain concepts. The process of creating useful classificatory categories by concept learning generally a a much harder process than using an existing classification structure. Thus, in medicine, discovery of a disease, *i.e.*, creation of a new category, is a relatively major event while diagnosis is much more routine. How these classificatory categories are created is an issue in research on learning and deep cognitive models [34]. In this paper we will deal only with the process of assigning an entity to an existing category in a classification structure.

IV. NUMERICAL APPROACHES TO CLASSIFICATION

So far we have discussed what is classification and why is it useful, but not how classification is accomplished, *i.e.*, we have presented the forms of input and output information for the classification task, and have provided an explanation for the usefulness of classification as a strategy, but have not presented any mechanism for performing the task. In the remainder of this paper we will review various knowledge-based, pattern recognition, and connectionist approaches to classification. In this section we will discuss numerical parameter setting approaches to classification. In the next section we will show how the use of intermediate abstractions reduces the computational complexity of performing the classification task, and discuss why symbols may be used to capture these abstractions. In section VI. we will discuss the use of syntactic and structural relations between symbols for classification, and in section VII. we will provide a detailed account of how complex symbolic structures that explicitly incorporate domain knowledge may be used for classification.

A. Statistical Pattern Recognition

Most early pattern recognition models used the statistical approach to classification [13] in which the object of unknown classification is represented as a multidimensional pattern vector. Each dimension of the vector represents an *attribute* of the entity, and typically is represented as a numerical variable, even though ordinals are some times used. The choice of the attributes of the entity is such that they have the potential to *distinguish* between the categories, where each category is characterized by some kind of probability distribution. In the task domain of medical diagnosis, *e.g.*, if it is desired to distinguish between diseases D_1 and D_2 and the system designer has reason to believe that symptoms $s_1, s_2 \ldots s_n$ carry useful information for this discrimination, then often careful statistical data gathering is possible such that a *discriminant function* of the variables $s_1, s_2 \ldots s_n$ is a very accurate classifier. When the number of dimensions is small, it is possible to design statistical classification systems that outperform human performance, since human reasoning with the same number of variables may be less efficient in information extraction. Despite the enormous intrinsic interest in the mathematical problem of designing classification algorithms in the discriminant function framework, Kanal and Chandrasekaran [24] have pointed out that the real computational power often comes from a careful choice of the attributes based on a good knowledge of the domain, rather than from the specific design of the separation algorithm.

What happens when the dimensionality of the pattern vector becomes very large, or the number of categories becomes large? When the number of categories increases, then in order to make more and more distinctions, generally the number of measurements on the entity of interest, *i.e.*, the dimensionality of the pattern vector, also needs to grow rapidly. The computational complexity of the algorithm to make the discrimination grows even more rapidly than the increasing number of dimensions, and correspondingly, the average performance, *i.e.*, the correct classification rate, deteriorates quite rapidly. Sensitivity problems become quite severe, *i.e.*, the required precision of the variables in the classification algorithm becomes impractically high. Opacity problems result, *i.e.*, it becomes increasingly hard to make any kind of statement about what attributes are playing what role in the recognition process. Szolovits and Pauker [40], discuss these and some of the other problems with probabilistic approaches to classification.

B. The Perceptron Model

Roughly in parallel with the development of statistical approaches to classification in the pattern recognition paradigm came the development of the early connectionist models of classification, specifically, the perceptron model. The perceptron architecture [31], consists of a set of input units and an output unit, each unit being a two-state, linear threshold digital device. Each unit in the input layer is connected directly to the output unit, with some numerical weight associated with each such connection. The inputs to the perceptron are points in an orthographic projection of the object to be classified, where each input unit scans some points in the projection. The output is the truth value of some predicate such as the predicate stating that the object, o_x, of unknown classification belongs to some known category, C_Y. The numerical weights associated with the connections in the network act as parameters of the network, and collectively represent the discriminant function for classification of the input object onto different categories. The output of the network is computed by a *linear combination* of the evidence that flows into the output unit *via* the connections. The perceptron architecture can be trained to "learn" the discriminant function by appropriately adjusting the weights of the connections in the network. Feedback on whether the network has reached the correct classificatory conclusion is provided by the trainer during the learning sessions. It has been shown that if the input objects are *linearly separable* then the weights of the connections will converge to the discriminant function that can correctly distinguish between the objects in finite time.

When the number of categories and the number of points scanned on the objects to be classified are small then the perceptron can be powerful classifier, at least for linearly separable objects. However, when these numbers get larger then the perceptron suffers from problems similar to those in the statistical approaches to classification. As the number of categories increases, the number of points needed to be scanned by the input units for learning the discriminant function increases, which results in a rapid increase in the number of input units. The time complexity of learning the right weights for correct classification grows even more rapidly, and correspondingly, the correct classification rate drops rapidly for a fixed number of input units. The sensitivity problem worsens, *i.e.*, even slight errors in the weights of the connections may result in large changes in the output. The opacity problem, *i.e.*, recognizing specifically which weight is playing precisely what role in the classification process, hard in the perceptron model in any case, becomes even harder. Minsky and Papert [28] discuss the computational properties of the perceptron architecture, and point out some of the problems with it.

V. USE OF INTERMEDIATE ABSTRACTIONS IN CLASSIFICATION

The above discussion shows that while numerical parameter setting schemes may lead to powerful classifiers for small problems, the complexity of the separation algorithm becomes impractically high as the number of classificatory categories increases. The problem here lies not so much in the specific choice of one discriminant function over another, but in the fact that these approaches seek to *directly* map the input entity onto classificatory categories. Indeed, similar complexity problems arise for *all* approaches that perform classification by directly mapping specific entities onto general categories. Let us consider, as another example of such direct classification, the method of discrimination tree traversal for medical diagnosis. Again, let the input be characterized by *n* state variables, s_1, s_2,..., s_n, each of which can take on one of *q* values. The state variables are organized in a tree in which the top node corresponds to some state variable s_1 and has q branches coming out of it, one for each of the q possible values that s_1 may take. The branches lead to q different nodes, each of which corresponds to some s_2 and has q branches coming out of it. This organization is repeated until all the state variables have been represented on the tree. Each of the q^n branches coming out of the q^{n-1} nodes at the n^{th} level leads to one of a finite number of disease categories, D_1, D_2,..., D_m. The time and space complexities for classification by discrimination tree traversal are given by $O(n)$ and $O(q^n)$, respectively [17]. Clearly, for complex, real world problems, where the number of classificatory categories typically is large, the proposition of directly mapping input entities onto classificatory categories is quite futile.

What, then, can be done when the number of classificatory categories is large? Let us consider, as an example, the problem of automatic reading of texts in some language that consists of a large number of words. Intuitively, one would think that first recognizing characters (or perhaps substrings of characters) in the words, and then recognizing word themselves would be computationally more attractive. The words (or perhaps word phrases) may be later used in understanding complete sentences in the language. In this approach, instead of performing classification by a direct mapping from the input entity onto the categories, intermediate abstractions are first constructed, the entity of unknown classification mapped onto these abstractions, which are then used as inputs to a higher-level classification process. What we are suggesting here is a *conceptual decomposition* of the classification process onto *hierarchically organized intermediate abstractions*. Such a conceptual decomposition makes the classification process more efficient, as we will see a little later.

A. Signature Tables

In order to make the notion of conceptual decomposition of the classification process into hierarchically organized intermediate abstractions more explicit, let us consider *evaluation functions* in game playing, *e.g.*, playing chess, as another example of classification. These functions usually yield a number which is a measure of the "goodness" of the board. For most purposes, effective use of this information can be made if the goodness is classified into one of a small number of categories. One of the first forms proposed for the evaluation functions was a linear polynomial of attributes of the board, where both the attributes and their weights were chosen in consultation with domain experts. Later, in order to take into account interactions between the variables in the evaluation function, higher order polynomials were proposed. This of course resulted in a fairly rapid increase in the complexity of the function: if r^{th} order interactions between the attributes were to be included, and the number of attributes is n, then the number of terms was of the order of n^f. Samuel's *signature tables* [33] provided a solution which exemplifies the use intermediate abstractions in classification. For the purposes of our discussion, Samuel's method can be described as follows:

1. Identify groups of attributes such that on the basis of domain knowledge there is reason to believe that they contribute to an intermediate abstraction that can be used to construct the desired classification, which in this case is a measure of the goodness of the board. The number of attributes in each group is kept small, and the attributes in a group may have some dependencies and interactions, in order to capture which polynomial terms were included in the more traditional evaluation functions. The abstractions typically correspond to the concepts in the task domain, *e.g.*, in chess, "defensibility of king" and "material advantage" may be such intermediate concepts, each of which can be estimated by a small subset of board attributes, while the final decision about the goodness of a board configuration may be made in terms of these intermediate abstractions.

2. Find a method of *classifying* the desirability of these intermediate concepts into a small number of categories from the values of the attributes in each group. The exact method for this classification is not especially important here, though Samuel proposed a specific mechanism for it. The essence of his mechanism is a mapping from a multidimensional vector, each component of which can only take on one of a small number of distinct values, to a symbolic abstraction, which can also take on one of a small number of distinct values. This mapping may be performed by a simple table look-up for example.

3. The outputs of the classifiers for each group can themselves be thought of as *qualitative* attributes at the next level of abstraction. These attributes can be then grouped and abstracted into higher level concepts, and the process repeated as many times as necessary, with only a small number of attributes in a group at any level, until the top-level concept is a classification of the "goodness" of the board.

Let n denote the total number of attributes at the lowest level of abstraction. Let us assume that the number of attributes in each group at any level in the hierarchy of abstractions is smaller than some small, constant, upper bound n_0 (an assumption allowed in the signature table method), and further, that the groups of attributes at any level are disjoint. Then both the time and space complexities are *O(n)* [17]. Even if a few attributes at some level are used in more than one group of attributes, which sometimes is the case, and in which case the time complexity would be somewhat worse than linear in *n*, clearly, the use of intermediate abstractions in classification yields substantial computational savings. Again, we are not suggesting that such conceptual decomposition of the classification process into hierarchically organized intermediate abstractions is always possible, but that, whenever possible, it is computationally advantageous to do so.

B. Hidden Units in Connectionist Networks

The computational power of using intermediate abstractions is evident from the fact that a major difference (perhaps *the* major difference) between modern connectionist networks and the perceptron model, is that the former provide mechanisms for capturing intermediate abstractions. In the perceptron model, since the input units were connected directly to the output unit, there was no representational mechanism to capture intermediate abstractions, and classification was performed by directly mapping input objects onto categories. Modern connectionist networks, on the other hand, contain hidden units between the input and the output units, thus providing a mechanism for representing intermediate abstractions as patterns of activity over the hidden units. The notion that the real role of the hidden units is to somehow capture these abstractions becomes clear from the following observation: in most connectionist schemes, such as the one for learning the past tenses of English language words [32], the *number of hidden units in the network is critical to its performance.* When the number of hidden units is too small then the problem is overconstrained and there is not enough structure to capture all the needed abstractions, as a result of which the performance of the network deteroriates markedly; and when the number of hidden units is too large then the problem is underconstrained and generalizations to the abstractions are not possible, again resulting in a marked deteroriation in the network performance. One method of handling these sensitivity problems is to make the number of hidden units a parameter of the architecture, and then experiment with the value of this parameter until the number of hidden units in the network is just right.

The real computational power of modern connectionist networks is thus based on the use of intermediate abstractions, which is an important reason for the resurgence of the connectionist paradigm in AI more than a decade after Minsky and Papert had showed the inadequacies of the perceptron model. Classification in connectionist architectures is accomplished by first mapping the input entity onto classificatory abstractions, and then mapping these abstractions onto output categories. Moreover, as in Samuel's work on signature tables for game playing programs, in modern connectionist networks the intermediate abstractions can be organized hierarchically. Indeed, for large scale connectionist networks, where the number of classificatory categories and intermediate abstractions may be very large, hierarchicalization of abstractions is an important method for dealing with the complexity of learning classificatory categories and intermediate abstractions [2].

C. Symbols and Abstractions

While the intermediate abstractions are represented as patterns of activity over the hidden units in connectionist networks, there is simpler way of capturing these abstractions: by means of discrete *symbols*. The representation of abstractions by symbols entails a trade off between the precision of numbers, with the concomitant problems of complexity, sensitivity, and opacity, for the *simplicity, flexibility, and perspicuity* of symbols. Often numbers are too precise for the task at hand, and robust symbolic hierarchical abstractions of the appropriate kind can capture almost all of the relevant information. These advantages of representing abstractions by symbols have been demonstrated most recently by Lehnert [25]. She has constructed a connectionistically inspired system, called PRO, for the task of word pronunciation, the same task that is performed by the entirely connectionist MBRtalk system. The main difference between the two approaches lies in that the PRO system uses symbols for capturing intermediate abstractions in the classification of character substrings of words. While PRO appears to perform at least as well the MBRtalk system, it is simpler, smaller, more robust, and more perspicuous. We are not suggesting that intermediate abstractions are entirely neutral to the underlying architecture of implementation and representing abstractions symbolically is necessarily right for all tasks. Chandrasekaran *et al.* [11] provide an analysis of the interaction between the abstractions needed for problem solving and the architecture for their implementation, and suggest that connectionist schemes may be well suited for simple forms of pattern matching and data retrieval, and for low-level parameter learning. However, for capturing higher level cognitive processes the advantages of using symbols for representing abstractions are just too important.

VI. USE OF RELATIONS BETWEEN SYMBOLS FOR CLASSIFICATION

After about a decade of work on statistical classification in the pattern recognition paradigm, during which work on classification in the perceptron and the symbolic paradigms was going on roughly in parallel, Narasimhan [29] proposed a *syntactic approach* to pattern classification. The idea was to describe categories of patterns not in terms of probability distributions in multidimensional spaces, nor in terms of intermediate abstractions that can be captured symbolically, but in terms of *relations between symbols*, much as grammatical categories are described in linguistic analysis. The idea of syntactic pattern recognition is really a special case of the more general notion of *structural relations* for describing classificatory categories. Thus, even when the idea of *syntax* is not appropriate --- it is doubtful that the notion of a picture grammar really is as general for the domain of visual objects as it appears from a purely formal perspective --- the notion of structural relations for characterizing categories may still be applicable. We note that the ability to describe a category in terms of relations is a move towards *descriptions* as the basis for category characterization.

The major research directions in pattern recognition for capturing structural relations generally were *formal*, *i.e.*, they used some or the other mathematical system within which theorems about relationships between categories may be provable regarding the classification performance. In fact, this was the major reason for the original emphasis on syntactic methods, since there was a well developed theory of formal grammars already available. This emphasis on formalisms led to two constraints: firstly, often an attempt was made to force the available formalisms to fit the pattern recognition problem, generally with unsatisfactory results; and secondly, because human classification performance was more heuristic in nature, restricted formalisms *could capture the quality of human performance* only fleetingly.

It is interesting to note that in connectionist schemes also classification is based on structural relations between *intermediate abstractions, even though the abstractions are represented* by patterns of activity over hidden units instead of being captured symbolically. The structural relations themselves are represented by connections of various types between the hidden units. Thus, in the MBRtalk system, the connectionist scheme for the task of word pronunciation, classification of the input words is based on the "syntactic relations" between the non-symbolic classificatory abstractions [37].

With the introduction of syntactic/structural relations between intermediate abstractions the progression of approaches to classification becomes

numbers ---> abstractions (symbols) ---> relations.

Now, if one is to use relations between symbolic attributes as the basis of category characterization, then why restrict oneself to *syntactic* relations? Why not bring the full power, to the extent possible or necessary, the *semantics* of the classificatory categories? Asking this question prepares the way for the next step in the progression of approaches to classification.

VII. KNOWLEDGE-BASED APPROACHES TO CLASSIFICATION

It is clear that each AI paradigm emphasizes different issues and poses them in a different language, *e.g.*, the pattern recognition paradigm raises issues such as those of discriminant functions, probability distributions, and error rates, while the connectionist paradigm raises issues such as those of weights of connections, hidden units, and parameter learning. Similarly, the knowledge-based reasoning paradigm focuses on the issues of how to *represent* knowledge in symbolic form, how to *organize* and *access* this knowledge, how to *use* this knowledge for solving problems, and how to *control* the problem solving process. The knowledge-based approaches to the classification task attempt to answer these questions for classificatory problem solving. In this section, we will describe *hierarchical classification* [6], [20] as an example of knowledge-based approaches to classification, using the task domain of medical diagnosis for illustration.

A. Hierarchical Classification

In hierarchical classification, domain knowledge is organized as a hierarchical collection of categories, each of which has knowledge that helps it determine its relevance to the input case of unknown classification. A fragment of the classification hierarchy for medical diagnosis might be as shown in Figure 1. Each category in the diagnostic classification hierarchy is a diagnostic *concept* of potential relevance to the case at hand. More general concepts (*e.g.*, LIVER) are higher in the hierarchy, while more particular ones (*e.g.* HEPATITIS) are lower in the structure.

The total diagnostic knowledge is *distributed* over the conceptual categories in the hierarchy. Each concept has "how-to" knowledge for simple evidential reasoning in the form of several clusters of *diagnostic rules*: confirmatory rules, exclusionary rules, and perhaps some recommendation rules. These production rules are of the form: <pattern> -----> <evidence>, *e.g.*, "If the value of SGOT is high then add *n* units of evidence in favor of cholestasis", where *n* is some small integer. The number of rules in any one cluster is kept small, and the evidence for confirmation and exclusion is suitably weighted and combined to arrive at a conclusion to establish or reject the relevance of the category to the case, or perhaps to suspend the decision making if there is not sufficient data to make a decision at the present time. The recommendation rules are optimization devices whose discussion is not necessary for our current purpose. What is important here is that when a concept in the classification hierarchy is properly

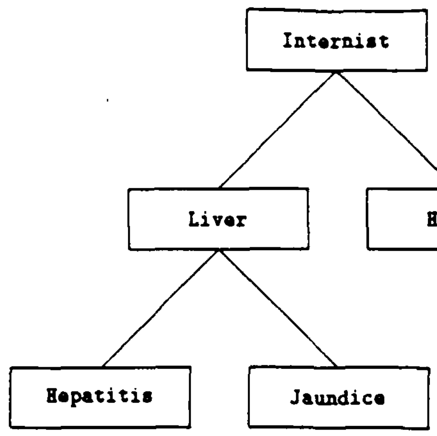


Figure 1: Fragment of a diagnostic classification hierarchy

invoked, a small, body of knowledge relevant for decision making comes into play.

The *control problem* in hierarchical classification can be stated as "which conceptual category should be considered at what point in the problem solving?". In general, we would like to use domain knowledge to achieve computational efficiency by considering only a subset of all categories. Similarly, we would like to consider categories which are more promising ahead of others. The control regime natural to hierarchical classification is top-down and can be characterized as *establish-refine*. Starting from the root node, each concept first uses its knowledge to establish or reject itself for relevance to the entity to be classified. If it succeeds in establishing itself, then it attempts refinement by *sending messages* to its subconcepts who repeat the establish-refine process. If, on the other hand, the concept rejects itself, then all its subconcepts are automatically ruled out leading to a pruning of the hierarchy. The idea is to establish a conceptual category, as specific as possible, that is relevant to the input entity. Let us consider the case of a patient suffering from hepatitis as an example. Given data about this patient, first INTERNIST would establish that there is in fact a disease, and send messages to LIVER and HEART for refinement as shown in Figure 1. Then LIVER would establish that the disease is a liver disease, and send messages to HEPATITIS and JAUNDICE for refinement, while HEART would reject the hypothesis that the patient is suffering from a heart disease. Next, HEPATITIS would establish the disease as hepatitis while JAUNDICE would rule out the hypothesis that the disease is jaundice. Thus each concept makes decisions about its relevance to the patient data in the *context* of the decisions made by its superconcepts. Sticklen *et. al.* [38] discuss the control issues in classificatory diagnosis in detail.

The problem solving in this approach to classification is *distributed.* The conceptual structures in the hierarchy are not a static collection of knowledge; instead, they are active problem-solving agents. Each of them has knowledge only about establishing or rejecting the relevance of a conceptual category, and communicates with others by passing messages. The entire ensemble of these semi-autonomous problem solving agents cooperates to perform the classification task. Goel *et al.* [19] have shown how the concurrency inherent in hierarchical classification can exploited on a distributed memory, message passing architecture.

We note that hard probability numbers are nowhere used in diagnosis by hierarchical classification; what each problem solving agent computes are *qualitative* belief measures: "definitely present", "likely present",..."definitely absent". Moreover, the computation of the qualitative values is *localized* rather than based on some global probability calculus; each agent computes the qualitative measure for *its* concept using only its own knowledge but in the context of its superconcepts. Medical diagnosis appears to be an instance of the class of problems in which a numerical approaches, such as statistical pattern recognition, would have significant computational problems. In addition, it would pose considerable difficulty in acquiring knowledge in terms of probability distributions, at least for problems of large degree of complexity, while knowledge in the form required by hierarchical classification is often directly available from domain experts.

At our research laboratory we have used the hierarchical classification methodology to construct MDX [6], [8], [20], a medical diagnostic system for a class of liver diseases in internal medicine. The number of state variables, such as symptoms, signs, and laboratory values, describing a typical case that MDX can handle is in the hundreds, and the number of distinct conceptual categories in its diagnostic hierarchy is also close to hundred. MDX is a complex system that has been tested on a number of real world cases with a high match between its conclusions and that of human specialists. Recently, a more sophisticated version of the MDX system, called MDX2 [39], has been constructed in our laboratory.

Several concerns ought to be noted before using the hierarchical classification methodology to build knowledge-based classificatory problem solvers:

1. Not all classification problems are necessarily solved as hierarchical classification problems.

   Hierarchical classification requires that concepts in the task domain be available at several different levels of abstraction. While there are many real world domains that do satisfy this condition, not every domain need have this characteristic. There are other systems that perform classification, but without using the hierarchical point of view [1]. However, it may be better to use hierarchical classification whenever possible for reasons of computational efficiency. Let *m* be the number of categories at the *leaf nodes* of the classification hierarchy. Since the desired classification generally is one of these *m* categories, the time complexity of non-hierarchical classification is *O(m.t)*, where *t* is the time complexity of finding the relevance of a single category to the entity of unknown classification. If the number of state variables is *n*, and single category classification is performed using the signature table approach discussed earlier, then *t* is *O(n)*. In case of hierarchical classification, in the best case when all but one branch at each node in the hierarchy are ruled out, the time complexity is *O(log(m).t)*; and in the worst case, when every branch at each node is traversed, the time complexity is *O(m.t)*. Goel *et al.* [17] provide details of the complexity calculations for classificatory reasoning. It is clear, however, that even in the worst case, the complexity of hierarchical classification is no worse than the complexity of non-hierarchical classification, and the choice between them really depends on whether it is possible to construct a classification hierarchy in the task domain of interest.

2. The entity to be classified may have several leaf node categories simultaneously relevant to it, rather than just one leaf node category. In medical diagnosis, *e.g.*, a patient may have both "cirrhosis" and "portal hypertension" (which in the domain of liver diseases might be two of leaf nodes in the classification hierarchy), and in addition, the two diseases may be causally related. Such a situation is not uncommon in other domains as well, *e.g.*, in character recognition, the pattern to be classified may consist of be two characters touching each other rather than one single character. The hierarchical classification framework clearly can deal with such situations.

3. The classification hierarchy may be a "tangled" hierarchy, *i.e.* some concepts in the hierarchy may have more than one superconcept. Such a hierarchy may be "untangled" in the hierarchical classification framework by storing a copy of the concept in each tangled branch. This introduces redundancy in the storage of domain knowledge by the classification agent.

4. In general, multiple classification hierarchies may exist in the task domain, *e.g.*, in medical diagnosis there may be one classification hierarchy for infectious diseases, and another for liver diseases. In addition, the same category may exist in more than classification hierarchy, e.g., viral hepatitis is a conceptual category in the infectious disease hierarchy as well as in the liver disease hierarchy. This involves *coordination* among the classifications reached by the different classification modules. The MDX2 system contains several classification hierarchies, and provides a mechanism for handling such interactions between them.

5. The problem task may require not only classification of entities onto categories, but other problem solving *types* as well, *e.g.*, the diagnostic task often is functionally decomposable into the generic tasks of *knowledge-directed data abstraction*, and *abductive assembly of explanatory hypotheses* in addition to that of classification [9], [10]. This involves coordinating the actions of various problem solving modules performing different generic tasks and cooperatively solving diagnostic problem. The MDX system [8] contained modules for hierarchical classification and knowledge-directed data abstraction and provided mechanisms for communication between them. The MDX2 system [39] contains modules for knowledge-directed data abstraction and abductive assembly of explanatory hypotheses in addition to several hierarchical classification modules, and provides mechanisms for handling interactions between them.

6. The conceptual structure mechanism used in hierarchical classification is only one of the several possible methods for determining the relevance of a specific category to the entity of unknown classification. In the DART system [16], *e.g.*, the decision about the match of the category to the input data is done by using theorem-proving techniques. Alternatively, the classification category agents may make their decisions based on a causal knowledge of the domain [34]. The MDX2 systems uses such causal knowledge to derive the conceptual structure needed for category classification. In simple cases, it may be possible to use statistical pattern recognition methods for this purpose. Connectionist networks may be especially appropriate for the pattern matching operations required in simple evidential reasoning [11]. The point is that *how* the hypotheses are evaluated is somewhat independent of the flow of control for the classificatory task as such, even though for complex problems, a rich knowledge structure will be called for to make the decision about

how well a specific category matches the data for the case in hand.

VIII. CONCLUSIONS

We have noted that classification appears to be an ubiquitous information processing task underlying human thought processes. The reason for this is the significant computational advantages that arise from indexing stored action knowledge over *equivalence classes of the states of the world* rather than over the states of the world themselves. We have taken the reader through a progression of approaches to classification:

numbers ---> abstractions (symbols) ---> relations ---> knowledge structures.

Each stage in this progression gave added power in controlling computational complexity by matching the structure of the classifier to that of the task. At the knowledge level, the computational power comes from *task-specific control regimes* controlling access to *appropriate chunks of domain knowledge*. We motivated the discussion by using classificatory diagnosis as an example in various places, but the ideas are applicable more generally.

This paper can be viewed as a bridge-building activity between three research paradigms in AI: knowledge-based reasoning, pattern recognition, and connectionism. Classification has been a major concern in pattern recognition, and an important task performed by most knowledge-based systems as well as by many connectionist networks. Thus, the classification task provides a good place to understand some of the distinctions between the three research paradigms. For well-constrained classification problems with relatively small number of categories, the numerical functions and measures used in pattern recognition models and connectionist networks typically can provide powerful classifiers which often outperform human experts by extracting the last trace of information that discrete symbolic processes can only approximate. On the other hand for complex problems involving many variables and categories the symbolic knowledge-based approach trades off the optimality of the best functions in pattern recognition and in connectionism for computational tractability and better matching with human knowledge in the task domain. Our own research lies in the knowledge-based reasoning paradigm. Our approach has been to identify *generic tasks* other than that of classification, but with the similar characteristic of being a building block for intelligence. Chandrasekaran [7], [9], [10] provides an account of the repertoire of generic tasks that we have identified so far.

Many of the points made in this paper transcend the particular task of classification. In that sense, this paper can be thought of as an attempt to show the need for the emergence of symbolic structures for complex information processing transformations on representations. Cybernetics showed the power and usefulness of feedback and stability in understanding many control and communication problems. However, classical control theory is expressed in terms of numerical measures and functions. Learning and control in this framework involves parameter modification and signal propagation. The space over which parametric changes and numerical signals can provide control is quite limited. Symbolic models of the world provide greater leverage for change and control and still keep computational costs under control. Thus in biological information processing, symbolization seems to have occurred very early in evolution; Lettvin *et al.* [26] provide an account of how the early visual processing of the frog is symbolic. Once symbols were available as the language in which to perform information processing, thought eventually evolved into more and more complex symbol structures. Thus the discussio⁻ in this paper can be viewed as an intuitive account of the emergence and power of symbolic structures for complex information processing activities.

Acknowledgments

We are thankful to Dean Allemang for his comments on an earlier version of this paper. We are grateful for the support of the Defense Advanced Projects Research Agency, RADC Contract (F30602-85-C0010), and the Air Force Office of Scientific Research, grant (87-0090), during the preparation of this paper.

References

[1] J.S. Aikins. "Prototypical Knowledge for Expert Systems". *Artificial Intelligence* 20(2):163-210, 1983.

[2] D.H. Ballard. "Modular Learning in Neural Networks". In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987, pages 279-284.

[3] J. Bennet and R. Engelmore. "SACON: A Knowledge-based Consultant for Structural Analysis". In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, 1979, pages 47-49.

[4] B. G. Buchanan and E.A. Feigenbaum. "Dendral and Meta-Dendral: Their Applications Dimension". *Artificial Intelligence* 11(1-2):5-24, 1978.

[5] D.C. Brown and B. Chandrasekaran. "Knowledge and Control for a Mechanical Design Expert System". *IEEE Computer Magazine* 19(7):92-100, 1986.

[6] B. Chandrasekaran, S. Mittal, F. Gomez, and J.W. Smith. "An Approach to Medical Diagnosis Based on Conceptual Structures". In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, 1979, pages 134-142.

[7] B. Chandrasekaran. "Towards a Taxonomy of Problem-Solving Types". *AI Magazine* 4(1):9-17, 1983.

[8] B. Chandrasekaran and S. Mittal. "Conceptual Representation of Mèdical Knowledge for Diagnosis by Computer: MDX and Related Systems". In *Advances in Computers*, M. Yovits: editor, Academic Press, New York, 1983, pages 217-293.

[9] B. Chandrasekaran. "Generic Tasks in Knowledge-based Reasoning: High-Level Building Blocks for Expert System Design". *IEEE Expert Magazine* 1(3):23-30, 1986.

[10] B. Chandrasekaran. "Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks". In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987, pages 1183-1192.

[11] B. Chandrasekaran, A. Goel, and D. Allemang. "Connectionism and Information Processing Abstractions: The Message Still Counts More Than the Medium". To appear in *AI Magazine*, 1988.

[12] W. J. Clancey. "Heuristic Classification". *Artificial Intelligence* 27(3):289-350, 1985.

[13] R.O. Duda, and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley, New York, 1973.

[14] R.O. Duda, P.E. Hart, P. Barret, J. Gasching, K. Konolige, and R. Reboh. "Development of the Prospector Consultation System for Mineral Exploration". Technical Report, SRI International, Menlo Park, California, 1979.

[15] J.A. Feldman and D.H. Ballard. "Connectionist Models and Their Properties". *Cognitive Science* 6:205-254, 1982.

[16] M.R. Genesereth. "Diagnosis Using Hierarchical Design Models". In *Proceedings of the Second National Conference on Artificial Intelligence*, 1982, pages 278-283.

[17] A. Goel, N. Soundararajan, and B. Chandrasekaran. "Complexity in Classificatory Reasoning". In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987, pages 421-425.

[18] A. Goel, B. Chandrasekaran, and D. Sylvan. "JESSE: An Information Processing Model of Political Decision Making". In *Proceedings of the Third Expert Systems in Government Conference*, 1987, pages 78-87.

[19] A. Goel, J.R. Josephson, and P. Sadayappan. "Concurrency in Abductive Reasoning". In *Proceedings of the DARPA Knowledge-based Systems Workshop*, 1987, pages 86-92.

[20] F. Gomez, and B. Chandrasekaran. "Knowledge Organization and Distribution for Medical Diagnosis". *IEEE Transactions on Systems, Man, and Cybernetics* 11(1):34-42, 1981.

[21] J.J. Hopfield and D.W. Tank. "Neural Computation of Decisions in Optimization Problems". *Biological Cybernetics* 52:141-152, 1985.

[22] J.R. Josephson, B. Chandrasekaran, J.W. Smith, and M.C. Tanner. "A Mechanism for Forming Composite Explanatory Hypotheses". *IEEE Transactions on Systems, Man, and Cybernetics* 17(3):445-454, 1987.

[23] L. Kanal and B. Chandrasekaran. "Recognition, Machine Recognition, and Statistical Approaches". In *Methodologies of Pattern Recognition*, Academic Press, New York, 1969, pages 317-332.

[24] L. Kanal and B. Chandrasekaran. "On Linguistic, Statistical, and Mixed Patterns for Pattern Recognition". In *Frontiers of Pattern Recognition*, Academic Press, New York, 1972, pages 163-192.

[25] W. Lehnert. "Case-Based Problem Solving with a Large Knowledge Base of Learned Cases". In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987, pages 301-306.

[26] J. Lettvin, H. Maturana, H., W.S. McCulloch, and W. Pitts. "What the Frog's Eye Tells the Frog's Brain". In *Proceedings of the IRE*, 1959, 47:1940-1951.

[27] J. McDermott. "R1: A Rule-Based Configurer of Computer Systems". *Artificial Intelligence*, 19(1):39-88, 1982.

[28] M. Minsky and S. Papert. *Perceptrons*, Expanded Edition, MIT Press, Cambridge, 1988.

[29] R. Narasimhan. "Labeling Schemata and Syntactic Description of Pictures". *Information and Control* 7:151-179, 1964.

[30] H.W. Pople. "Heuristic Methods for Imposing Structure on Ill-Structured Problems". In *Artificial Intelligence in Medicine*, P. Szolovits: editor, Westview Press, Boulder, Colorado, 1982, pages 119-190.

[31] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[32] D.E. Rumelhart and J.L. McClelland. "On Learning the Past Tenses of English Verbs". In *Parallel Distributed Processing, Volume 1*, Rumelhart, McClelland and the PDP Research Group: editors, MIT Press, Cambridge MA, 1986.

[33] A.L. Samuel. "Some Studies in Machine Learning Using the Game of Chequers II: Recent

Progress". *IBM Journal of Research and Development* 11(6):601-617, 1967.

[34] V. Sembugamoorthy and B. Chandrasekaran. "Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems" In *Experience, Memory, Reasoning*, J. Kolodner and C. Reisbeck: editors, Lawrence Earlbaum, Hillsdale N.J., 1986, pages 47-73.

[35] E.H. Shortliffe. *Computer-based Medical Consultations: MYCIN*. Elsevier/North-Holland, 1976.

[36] R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy*. Freeman, San Francisco, 1963.

[37] C. Stanfill and D. Waltz. "Toward Memory-based Reasoning". *Communications of the ACM* 29(12):1213-1228, 1986.

[38] J. Sticklen, B. Chandrasekaran, J.R. Josephson. "Control Issues in Classificatory Diagnosis". In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pages 300-306.

[39] J. Sticklen. "MDX2: An Integrated Medical Diagnostic System". PhD. Dissertation, Department of Computer and Information Science, *The Ohio State University, 1987.*

[40] P. Szolovits and S.G. Pauker. "Categorical and Probabilistic Reasoning in Medical Diagnosis". *Artificial Intelligence* 11:115-144, 1978.

87-AG-DISSYNCOMP

The Ohio State University
Department of Computer and Information Science
Laboratory for Artificial Intelligence Research

Technical Report
October 1987

Distibuted Synthesis of Composite Explanitory Hypotheses

Ashok Goel, P. Sadayappan, and N. Soundararajan
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

Note: Submitted for publication

Distributed Synthesis of Composite Explanatory Hypotheses

Ashok Goel, P. Sadayappan, and N. Soundararajan

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

Abstract

Abductive reasoning has received much recent attention in Artificial Intelligence research on knowledge-based systems. The general abductive task is to infer a hypothesis that best explains a set of data. Typical subtasks of this are generating hypotheses that can account for various subsets of the data, and using these hypotheses as components in synthesizing a composite hypothesis that best explains the data set. In this paper, we present a model for distributed synthesis of composite explanatory hypotheses. We provide concurrent algorithms for synthesizing a composite hypothesis, and compare their time complexity with the sequential algorithms. The algorithms are specified in the language of Communicating Sequential Processes, and the model can be implemented on a distributed memory, message passing, parallel computer architecture.

1. Introduction

Abductive reasoning has received much recent attention in Artificial Intelligence (AI) research on knowledge-based systems [Pople, 1977; Reggia, 1983; Josephson *et al.*, 1987; Pearl, 1987]. The information processing task of abduction is to infer a hypothesis that best explains a set of data. A typical subtask of this is to generate hypotheses that can account for various subsets of the data. Another typical subtask is to use these hypotheses as components in synthesizing a composite hypothesis that best explains the data set. However, synthesizing composite explanatory hypotheses can be computationally very expensive, especially in the presence of certain types of interactions between the component hypotheses [Allemang *et al.*, 1987; Bylander *et al.*, 1987]. This suggests that abductive reasoning systems should exploit concurrency in synthesizing composite hypotheses.

We have elsewhere reported [Goel *et al.*, 1987], on a shared memory, "blackboard" model for concurrent synthesis of composite hypotheses. In this paper, we present a model for distributed synthesis of composite explanatory hypotheses that can be implemented on a distributed memory, message passing, parallel computer architecture. The main reason for this is that the current model for synthesizing composite hypotheses provides a more modular organization of processing, and a more "natural" synchronization mechanism between concurrently executing processes.

2. Abductive Reasoning

2.1. Abductive Inference

Abduction is a form of logical inference that may be characterized as follows [Josephson *et al.*, 1987]:

> *D* is a collection of data (facts, observations, givens).
> *C* is a hypothesis (one of possibly many hypotheses)
> *C* explains *D* (would, if true, explain *D*),
> No other hypothesis explains *D* as well as *C* does.
> ---
> Therefore, *C* is (probably) correct.

Abductive inference appears to be ubiquitous in knowledge using reasoning [Charniak and McDermott, 1985]. Abduction occurs, for instance, in diagnostic problem solving, where the data is in the form of symptoms, and the explanatory hypotheses are component malfunctions (or diseases) [Pople, 1977; Reggia, 1983; Sticklen, 1987]. Scientific data interpretation (where the data is in the form of sensor readings, and the explanatory hypotheses are about object structures), and military situation assessment (where the data is in the form of events, and the explanatory hypotheses are plans ascribed to the adversary), are also instances of abductive inference making. Some aspects of perception, and some aspects of natural language understanding, appear to be abductive in character as well.

2.2. Abductive Task and Subtasks

Our research on abductive inference takes place in the context of a theory of generic information processing tasks in knowledge using reasoning [Chandrasekaran, 1986; Chandrasekaran, 1987]. A generic task is a "natural kind" of information processing task, functionally specified by the input it takes and the output it gives. For each generic task, there exists a strategy characterized by the organization of knowledge, and the control of processing that it uses for performing the generic task computationally efficiently. Generic tasks, and their corresponding strategies, provide high-level building blocks for the design and construction of knowledge-based systems. If a complex real-world information processing task can be functionally decomposed into several generic tasks, and if we know of strategies for performing the generic tasks efficiently, then there is a basis for concluding that the complex task can be successfully performed by an integrated knowledge-based system.

An example of a generic task is the Hierarchical Classification generic task [Gomez and Chandrasekaran, 1984], which takes as input a set of data describing a specific case, and gives as output a set of hypotheses that can account for various subsets of the data with high *prima facie* belief values. Hierarchical Classification is performed by a computationally efficient strategy that uses a taxonomic hierarchical organization of the hypotheses, and a top-down control of processing. This strategy may be executed concurrently [Goel *et al.*, 1987]. The Abductive Assembly generic task [Josephson *et al.*, 1987], which takes as input hypotheses that can explain, with high belief values, various subsets of a data, and gives as output a composite hypothesis that best explains the data set, is another example of a generic task. We will describe sequential and concurrent mechanisms for performing the Abductive Assembly generic task a little later.

Under the assumption that domain knowledge is available in the appropriate forms, the abductive task may be functionally decomposed into the generic tasks of Hierarchical Classification of data, and Abductive Assembly of a composite explanatory hypothesis [Josephson *et al.*, 1987]. The main advantage of this decomposition is that classification of the data reduces the size of the hypothesis space that needs to be searched in assembling a composite explanatory hypothesis. Instead of searching the space of all hypotheses, the assembler needs to search only the space of hypotheses with high belief values. The RED system [Smith *et al.*, 1985], is an integrated knowledge-based system, for identifying red-cell antibodies for use in medical blood banks, that explicitly uses the classification and assembly mechanism for performance of a version of the general abductive task. The MDX2 system [Sticklen, 1987], an integrated knowledge-based system for diagnosis of a class of diseases in internal medicine, also uses the classification and assembly mechanism for performing a version of the abductive task.

3. Abductive Assembly of Composite Explanatory Hypotheses

3.1. Definitions

Let $D = \{d_i\}$, $i=1,2,\dots,n$ be a set of n observed data. Let $H = \{h_j\}$, $j=1,2,\dots,m$ be a set of m hypotheses that can explain various subsets of D with high *prima facie* belief values. Let e be a map from subsets of H to subsets of D; e: $2^H \rightarrow 2^D$. We may interpret $e(H_j)=D_i$, where $H_j \subseteq H$ and $D_i \subseteq D$, as the explanatory coverage of H_j, *i.e.* H_j can explain only and all members of D_i. Let V be a set of v discrete values. Let b be a map from H to V: $b : H \rightarrow V$. Each $h_j \in H$ has a belief value $b(h_j)$ from V assigned to it.

We may characterize abductive Assembly of composite explanatory hypotheses as a five-tuple $<D, H, e, b, C>$, where D, H, e, and b are as defined above, and constitute the input to the task; and C, the output of the task, is a subset of H, $C \subseteq H$, that best explains D. This characterization is incomplete since we have not yet characterized what is meant by a best explanation. Unfortunately, there is no commonly accepted definition of a best explanation. Operationally, a composite hypothesis, C, that "best" explains the data set, D, may be assembled based on the following three criteria.

- Complete explanatory coverage of data: A hypothesis C_1 is a better explanation of D than a hypothesis C_2, if $e(C_2) \subset e(C_1)$. Ideally, the assembled composite hypothesis, C, would provide complete explanatory coverage of D, *i.e.* $e(C)=D$.

- Maximal belief value of component hypotheses: If two composite hypotheses C_1 and C_2 have the same explanatory coverage of the members of D, then C_1 is a better explanation of D than C_2, if for each datum $d \in D$ that any $h_2 \in C_2$ explains, there exists a $h_1 \in C_1$ that can explain d, and h_1 has a belief value equal to, or greater than that of h_2.

- Parsimonious composite hypothesis: If two composite hypotheses C_1 and C_2 have the same explanatory coverage of the members of D, then C_1 is a better explanation of D than C_2, if C_1 is a proper subset of C_2, $C_1 \subset C_2$.

We note that there is no *a priori* guarantee that there exists a unique "best" explanation.

3.2. Generating Composite Explanatory Hypotheses

Let us postulate that the members of H are non-interacting, *i.e.* they are mutually compatible, and represent explanatory alternatives where their explanatory capabilities overlap. The task of the assembler is to construct, using the members of H as components, a "best" composite hypothesis, C, for explaining the members of D. The serial assembler of the RED system builds the composite hypothesis, C, using a specialized means-ends mechanism whose goal is a complete explanation of the members of D. The assembler detects differences between the goal state (all of D has been explained), and the present state (some $d \in D$ has not been explained). It then selects an $h \in H$ which can explain the unexplained d, and integrates this h into the growing composite hypothesis C.

3.3. Testing Composite Explanatory Hypotheses

Once the composite explanatory hypothesis, C, has been assembled, it may be tested for parsimony. A composite hypothesis is parsimonious if it has no explanatorily superfluous components, where a component hypothesis in C is explanatorily superfluous if removing it from C does not reduce the explanatory coverage of D. Starting with the hypothesis with the lowest belief value, each hypothesis in C may be tested for parsimony, and removed from C if it is explanatorily superfluous. After testing for parsimony, the composite hypothesis C may be tested for essentialness of component hypotheses. A hypothesis h in C may be tested for essentialness by temporarily removing it from H and reassembling a composite hypothesis. If there is no way to reassemble a composite hypothesis without reducing explanatory coverage of D, then the h is essential; otherwise it may be substituted by another hypothesis, and the composite hypothesis may be reassembled using the substitute hypothesis.

3.4. Interacting Component Hypotheses

So far we had assumed that the hypotheses in H are non-interacting. In fact, several distinct types of interaction are possible between two hypotheses $h_1, h_2 \in H$ [Josephson *et al.*, 1987]:

- Associativity: The inclusion of h_1 in C suggests the inclusion of h_2. Such an interaction may arise if the assembler has knowledge of, say, a statistical association between h_1 and h_2.

- Additivity: h_1 and h_2 cooperate additively where their explanatory capabilities overlap. This may happen if h_1 and h_2 can separately explain some datum $d \in D$ only partially, but collectively can explain it fully.

- Incomptability: h_1 and h_2 are mutually incompatible, *i.e.* if one of them is included in C then the other should not be included.

- Cancellation: h_1 and h_2 cancel the explanatory capabilities of each other in relation to some $d \in D$. For example, h_1 might imply that some data value will increase, while h_2 may imply that the value will decrease, thus canceling each other's explanatory capability with that datum.

The RED system accommodates the additivity, and pair-wise incompatibility interactions between component hypotheses.

3.3. Computational Complexity of Serial Assembly

Under the assumption that the hypotheses in H are non-interacting, the worst case time complexity of RED's algorithm for generating a composite explanatory hypothesis is given by,

$$T_{hypothesis\ generation}(n,m) = O(n(m+n\times log(n))) ,$$

where m is the cardinality H, and n is the cardinality of D [Allemang *et al.*, 1987]. Similarly, the worst case time complexity for testing the composite hypothesis for parsimony is given by,

$$T_{parsimony\ testing}(n,m) = O(m\times n\times log(n))$$

and the worst case time complexity for testing the composite hypothesis for essentialness of component hypotheses is given by,

$$T_{essentialness\ testing}(n,m) = O(m\times n\times (m+n\times log(n)))$$

Thus, for non-interacting component hypotheses, the task of Assembling a composite explanatory hypothesis is in the class of ***P*** problems. Abductive assembly of composite explanatory hypotheses remains in the class of ***P*** problems even in the presence of associativity and additivity types of interactions. However, in the presence of incompatibility or cancellation types of interactions the task in the class of ***NP-Hard*** problems [Bylander *et al.*, 1987].

4. Distributed Abductive Assembly of Composite Explanatory Hypotheses

4.1. Concurrency in Abductive Assembly

There are two types of questions that are raised during abductive assembly of a composite explanatory hypothesis. The first type is from the perspective of each $d_i \in D$, and is of the form "Which hypothesis $h_j \in H$ can best explain me?". This type of question can be asked and answered for each $d_i \in D$ independently of others. The second type of question is from the perspective of each $h_j \in H$, and is of the form "Which elements of D should I be used to explain?". Again, this type of question can be asked and answered for each $h_j \in H$ independently of others.

Let $P = \{p_i\}$, $i=1,2,...,n$ be a set of n processes, one for each $d_i \in D$, $i=1,2,...,n$. Each $p_i \in P$ process represents the perspective of the corresponding datum $d_i \in D$ during abductive assembly of a composite explanatory hypothesis. The p_i, $i=1,2,...,n$ processes use identical algorithms, and may execute concurrently. Similarly, let $Q = \{q_j\}$, $j=1,2,...,m$ be a set of m processes, one for each $h_j \in H$, $j=1,2,...,m$. Each $q_j \in Q$ process represents the perspective of the corresponding hypothesis $h_j \in H$, during assembly of a composite hypothesis.

Again, the q_j, $j=1,2,...,m$ processes use identical algorithms, and may be executed concurrently.

4.2. Distributed Generation of Composite Explanatory Hypotheses

In RED's mechanism for abductive assembly of composite explanatory hypotheses, first a composite explanatory hypothesis is generated, and then it is improved by testing for parsimony, and essentialness. However, in our model for assembly of composite hypotheses, the hypotheses essential for explaining some data are identified during the generation of the composite hypothesis itself. This eliminates the need for testing the composite hypothesis for essentialness of its component hypotheses, and generating new composite hypotheses in case the test fails. Moreover, identifying the essential hypotheses and the subsets of data that they can explain, reduces the size of unexplained data. This may reduce the time complexity of generating composite explanatory hypotheses.

In our model for distributed abductive assembly of a composite explanatory hypothesis, the n P processes, and the m Q processes can all be executed concurrently, *i.e.* $p_1 \,//\, p_2 \,//\, \ldots \,//\, p_n \,//\, q_1 \,//\, q_2 \,//\, \ldots \,//\, q_m$, where the symbol $//$ denotes concurrently executable processes. The information processing alternates between the P processes and the Q processes. In each cycle of processing, when the P processes are executing the Q processes are idle: when the P processes have finished executing, they communicate their results to the appropriate Q processes, and the Q processes can start executing. Similarly, when the Q processes are executing the P processes are idle; when the Q processes have finished executing, they communicate their results to the appropriate P processes, and the P processes can start executing. This cycle continues until the composite hypothesis has been fully assembled. Thus, the P and the Q processes contribute separately to the assembly of the composite hypothesis from the the data and the hypotheses perspectives, respectively.

At the start of processing, each process $q_j \in Q$ has information specifying the hypothesis $h_j \in H$ that it represents, the explanatory coverage e of the hypothesis, the belief value b_j of the hypothesis, and the data set D that is to be explained. This information may be posted by the hierarchical classifier(s). Similarly, each process $p_i \in P$ has information specifying the $d_i \in D$ that it represents, and the cardinality of the set H. Since the n P processes use identical algorithms, and the m Q processes also use identical algorithms, it suffices to describe the processing from the perspectives of a process $p_i \in P$, and a process $q_j \in Q$.

In the first cycle of processing the essential hypotheses are identified. The q_j process, representing some hypothesis $h_j \in H$, sends its belief value b_j to processes in P corresponding to the data in the explanatory coverage $e(h_j)$. The p_i process, representing some datum $d_i \in D$, receives the belief values of all hypotheses that can explain the d_i. From the perspective of the p_i, three things may happen.

1. p_i receives no messages. Then the d_i is <u>unexplainable</u>, and p_i does nothing.

2. p_i receives exactly one message. Then the hypothesis corresponding to the process in Q from whom p_i received the message is <u>essential</u>. p_i sends a message to that process in Q indicating this.

3. p_i receives more than one message. Then the hypotheses corresponding to the processes in Q from whom p_i received the messages are not essential. p_i sends a null message to these processes in Q.

The q_j process receives messages from processes in P corresponding to the data in $e(h_j)$.

In the second cycle of processing, hypotheses for explaining data that cannot be explained by the essential hypotheses are selected. From the perspective of q_j two things may happen.

1. q_j receives at least one message indicating that the corresponding hypothesis h_j is essential. Then q_j sends a message to processes in P corresponding to the data in $e(h_j)$, indicating that they can be explained.

2. q_j receives only null messages. Then q_j sends null messages to processes in P corresponding to the data in $e(h_j)$.

The p_i process receives messages from the processes in Q corresponding to the hypotheses that can explain the d_i. From the perspective of p_i, two things may happen.

1. p_i receives atleast one message indicating that the d_i can be explained by some essential hypothesis. Then p_i does nothing.

2. p_i receives only null messages. Then p_i selects from the hypotheses that can explain the d_i, the hypothesis with the highest belief value. If the belief values for two or more hypotheses that can explain the d_i are the same, then p_i selects a hypothesis based on its explanatory coverage. If that will not break the tie, then selection is made at random. On selection of a hypothesis, p_i sends a message to the corresponding process in Q indicating that the hypothesis should be included in the composite hypothesis. p_i also sends a null message to processes in Q corresponding to other hypotheses that can explain the d_i.

The q_j process receives messages from processes in P corresponding to the data in $e(h_j)$.

At the end of the second cycle, a composite hypothesis has been generated. The composite hypothesis contains all the essential hypotheses, and can explain as much of the data as is explainable. We have not as yet addressed the issue of synchronization of sending and receiving messages between the P and the Q processes. The framework of Communicating Sequential Processes (CSP) Hoare, 1978 , provides a synchronization mechanism between concurrently executing processes that is quite natural to distributed abductive assembly of composite explanatory hypotheses. CSP is a language for concurrent programming on distributed memory, message passing, parallel computer architectures. Indeed, concurrency is a primitive of CSP. Input and Output are also primitives of CSP. and are used for sending and receiving messages from one process to another. Communication occurs when one process names another as destination and the second process names the first as source. Synchronization between processes is achieved by delaying an input or output command until the other process is ready with the corresponding output or input. Nondeterminism in CSP is controlled by use of guarded commands [Dijkstra, 1975]. We provide algorithms for distributed generation of a composite explanatory hypothesis in the language of CSP in the Appendix.

4.3. Distributed Testing of Composite Explanatory Hypotheses

Once a composite explanatory hypothesis has been assembled as shown above, it may be tested for parsimony. However, in general there appears to be no concurrent mechanism for testing the composite hypothesis for parsimony with time complexity better than that for the serial mechanism. Testing the composite hypothesis for parsimony can be performed concurrently only when there is no overlap between the explanatory coverages of the inessential component hypotheses in the composite hypothesis. In that case, the q_j process corresponding to some inessential hypothesis h_j in the composite hypothesis, may send a message to processes in P corresponding to the data in $e(h_j)$. The p_i process corresponding to a datum d_i that can be explained only by an inessential hypothesis, may decide if some other component hypothesis in the composite hypothesis can explain the d_i, and if so, sends a message to the appropriate processes in Q, indicating that the previously selected hypothesis is explanatorily superfluous. The q_j process corresponding to the previously selected hypothesis h_j, may now remove the h_j from the composite hypothesis.

In general, explanatory coverages of inessential component hypotheses in the composite hypothesis will overlap. In that case, the concurrent mechanism for testing a composite hypothesis for parsimony outlined above, may leave some explainable datum unexplained. The fact that in general there appears to be no concurrent mechanism for testing the composite hypothesis for parsimony with time complexity better than that for the serial mechanism, without leaving some explainable datum unexplained, may indicate that intelligent agents in routine situations typically do not test composite hypotheses for parsimony because it can be computationally expensive. Instead, intelligent agents may invest their computational resources in testing of composite explanatory hypotheses for parsimony only in special situations such as medical diagnosis, where it may be especially important to do so.

4.4. Accommodating Interactions in Distributed Abductive Assembly

So far we had assumed that the hypotheses in H were non-interacting. In fact, the distributed assembler, can accommodate associativity, additivity, pair-wise incompatibility, and pair-wise cancellation types of interactions. We will not describe here the mechanisms for accommodating these interactions due to lack of space; indeed, that is the subject matter of another paper. However, as an example we will outline how the distributed assembler accommodates associativity interactions between component hypotheses. We recall that associativity interactions occur when the inclusion of some hypothesis h_1 in C suggests the inclusion of some other hypothesis h_2. In distributed assembly, if some hypothesis h_1 is included in the composite hypothesis, and if the corresponding process q_1 has knowledge of an association between h_1 and some other hypothesis h_2, then q_1 sends a message to the q_2 process corresponding to h_2, indicating that h_1 has been included in the composite hypothesis, and that h_2 should also be included. On receiving this message, q_2 includes h_2 in the composite hypothesis. In this way the associativity interaction between the component hypotheses h_1 and h_2 is accommodated.

4.5. Computational Complexity of Distributed Assembly

Under the assumption that the hypotheses in H are non-interacting, the worst case time complexity for distributed generation of a composite explanatory hypothesis is given by,

$$T_{hypothesis\ generation}(n,m) = O(n+m)$$

where n is the cardinality of D, and m is the cardinality of H. Since the essential hypotheses were identified while generating the composite explanatory hypothesis, the time complexity of testing the composite hypothesis for essentialness of component hypotheses is already included in the time complexity of generating the composite hypothesis. In case there is no overlap between explanatory coverages of inessential component hypotheses in a composite explanatory hypothesis, the worst case time complexity of testing the composite hypothesis for parsimony is given by,

$$T_{testing\ parsimony}(n,m) = O(n \times m)$$

We note that the constants in the time complexities for serial, and distributed generation of composite hypotheses are comparable, since they arise from linear search in both cases. In order to fully compare the time complexities of serial and distributed models of generating composite explanatory hypotheses, we need an estimate of the values of n and m. However, the values of n and m vary from domain to domain, and even from case to case. In the domain of the RED system, for a typical case the values may be *40* for n, and *15* for m. Thus, distributed abductive assembly of composite explanatory hypotheses may provide significant speed up of processing over serial assembly.

However, for several reasons we wish to be cautious about this claim. Firstly, the time complexities that we have given are for the worst case, and not for the "average" case since the "average" case is so domain dependent. Secondly, in general the time complexity of concurrent testing of composite hypotheses for parsimony is no better than that of serial testing. Thirdly, the time complexities for serial and distributed assembly of composite explanatory hypotheses that we have given are valid only under the assumption that the hypotheses in H are non-interacting. The distributed assembler can accommodate the associativity, additivity, pair-wise incompatibility, and pair-wise cancellation interactions. However, the general problem of assembling composite explanatory hypotheses in the presence of incomptability and cancellation interactions between the hypotheses in H is in the class of ***NP-Hard*** problems. Finally, we have not accounted for the costs of communication between the P and the Q processes in the time complexity for the distributed assembler. Even if we assume that n (typically n is greater than m) channels for communication between the n P and the m Q processes are available, the communication overhead costs could be significant.

5. Conclusions

Abductive inference appears to be ubiquitous in knowledge using reasoning. However, the task of assembling composite explanatory hypotheses, a subtask of the general abductive task, can be computationally very expensive. This poses a dilemma: how to construct computationally efficient knowledge-based systems for abductive reasoning? We have provided a model for distributed assembly of composite explanatory hypotheses, based on the framework of communicating sequential processes. In our model, a process is associated with each datum, and with each hypothesis. The data processes and the hypothesis processes are concurrently executable. Abductive assembly of a composite hypothesis is viewed from multiple perspectives (the data perspective and the hypotheses perspective), with alternation between the perspectives. Each alternation produces intermediate results, which are unified to obtain the composite hypothesis. We showed that distributed generation of composite hypotheses may provide significant speed up of processing over serial generation. In addition, the essential hypotheses can be identified while generating composite hypotheses. However, testing of a composite hypothesis for parsimony in general appears to be inherently sequential. We suggested that this model can accommodate different types of interactions between component hypotheses. The model can be implemented on a distributed memory, message passing, parallel computer architecture.

Appendix

The concurrent algorithms for distributed generation of composite explanatory hypotheses given below are from the perspectives of the processes q_j and p_i, respectively, and are in the language of CSP. We assume that the "Cons" cell with two elements, a head and a tail, is a data object in CSP. We assume also that "Cons" is a primitive function of CSP for constructing a Cons cell given a head and a tail element, and that "Left" and "Right" are primitive functions that give the head and the tail elements of a Cons cell, respectively. In the algorithms below we will use the symbols ";" as the command delimiter, "□" as the the guarded command separator, and "/*" and "*/" as comment delimiters.

0.1. Algorithm for the q_j Process

```
q_j:: /* The process q_j represents the perspective of
   hypothesis h_j */

n:integer; /* Given number of processes in P */
d:(1...10)character; /* Contains name of some datum */
ToBeExplained:(1...n)d; /* Given array of the data to be explained */
n_j:integer; /* Given number of datum that h_j can explain */
ICanExplain:(1...n_j)d; /* Given array containing all
   d_i ∈ D that the h_j can explain */
b_j:integer; /* Given belief value of the hypothesis h_j */
x:(1...10)character; /* Dummy variable */
y,k1,k2:integer; /* Dummy variables */
Status,MyStatus:(1...10)character; /* Status is a dummy variable; MyStatus
   contains information about the current status of h_j */

/* Send the belief value b_j of the hypothesis h_j, to each
   processor p_i ∈ P corresponding to d_i ∈ D
   that the h_j can explain; send the value zero to all other
   processors in P */
k1:=1;
k2:=1;
*[ k1≤n→
    x:=ToBeExplained(k1);
    y:=0;
    *[ k2≤n_j→
        [ICanExplain(k2)≠x→skip;
        []ICanExplain(k2)=x→y:=b;
        ]
        k2:=k2+1;
    ]
    P_x!y;
    k1:=k1+1;
]
/* Receive message on whether the h_j is Essential, and if so, then set
   MyStatus to Essential */
k1:=1;
*[ k1≤n_j→
    x:=ICanExplain(k1);
    P_x?Status;
    [Status≠Nil→skip;
    []Status=Essential→
        MyStatus:=Essential;
    ]
    k1:=k1+1;
```

```
/* If the $h_j$ is Essential, then send a Explained message to each
   $p_i \in P$ corresponding to $d_i \in D$ that the
   $h_j$ can explain */
MyStatus=Essential→
   k1:=1;
   * k1≤n_j →
      x:=ICanExplain(k1);
      Status:=Explained;
      P_x!Status;
      k1:=k1+1;

/* If the $h_j$ is not Essential, then send a Nil message to
   $p_i \in P$ corresponding to $d_i \in D$ that the
   $h_j$ can explain */
◇MyStatus≠Essential→
   k1=1;
   * k1≤n_j →
      x:=ICanExplain(k1);
      Status:=Nil;
      P_x!Status;
      k1:=k1+1;

/* Further, if the $h_j$ is not Essential, then receive message on
   whether the $h_j$ should be included in the composite hypothesis; if
   so, set MyStatus to In */
   k1:=1;
   * k1≤n_j →
      x:=ICanExplain(k1);
      P_x?Status;
      Status=Nil→skip;
      ◇Status=In→MyStatus:=In;

      k1:=k1+1;
```

0.2. Algorithm for the p_i Process

```
$p_i$:: /* The process $p_i$ represents the perspective of datum $d_i$ */

m:integer; /* Given number of processes in $Q$ */
x, Best:(1...10)character; /* Dummy variables */
y, k1, k2, k3:integer; /* Dummy variables */
z:Cons(x,y); /* The head element contains some hypothesis $h_j$ that can
   explain the $d_i$, and the tail element contains its belief value $b_j$ */
CanExplainMe:(1...m)z; /* A constructed array of Cons cells containing
   information about hypotheses that can explain $d_i$; the head element
```

```
    of each Cons cell contains some $h_j$ that can explain the $d_i$,
    and the tail element contains the belief value with which the $h_j$
    can explain it *)
Status,MyStatus:(1...10)character; (* Status is a dummy variable; MyStatus
    contains information about the current status of the $d_i$ *)

 (* Receive messages as to which all $h_j \in H$ can explain the $d_i$ *)
k1:=1;
k2:=0;
*[ k1≤m→
      Q_{k1}?y;
      [ y=0→skip;
       □ y≠0→
          k2:=k2−1;
          z:=Cons(k1,y);
          CanExplainMe(k2):=z;
      ]
      k1:=k1−1;
 ]
(* If no $h_j \in H$ can explain the $d_i$, then the $d_i$
    is Unexplainable *)
[k2=0→
    MyStatus:=Unexplainable;
 (* If only one $h_j \in H$ can explain the $d_i$, then the $h_j$ is Essential *)
 □ k2=1→
    x:=Left(CanExplainMe(k2));
    Status:=Essential;
    Q_x!Status;
 (* If more than one $h_j \in H$ can explain the $d_i$, then send
    a Nil message to each $q_j \in Q$ corresponding to
    $h_j \in H$ that can explain the $d_i$ *)
 □ k2>1→
    k1:=1;
   *[ k1≤k2→
        x:=Left(CanExplainMe(k1));
        Status:=Nil;
        Q_x!Status;
        k1:=k1−1;
    ]
]
  (* Receive messages from $h_j \in H$ that can explain the
    $d_i$ on whether the $d_i$ has been explained by some hypothesis,
    if so, set MyStatus to Explained *)
 k1:=1;
   *[ k1≤k2→
        Q_{k1}?Status;
```

```
          Status=Nil→skip;
          ◇Status=Explained→
             MyStatus:=Explained;

          k1:=k1+1;

   * If no Essential $h_j \in H$ can explain the $d_i$, then send a
     In message to the $q_j \in Q$ corresponding to the
     $h_j \in H$ that can best explain the $d_i$, and a Nil
     message to $q_j \in Q$ corresponding to all other
     $h_j \in H$ that can explain the $d_i$ *
   MyStatus=Explained→skip;
   ◇MyStatus≠Explained→
      k1:=1;
      Best:=Left(CanExplainMe(k1));
      k3:=Right(CanExplainMe(k1));
      k1:=k1+1;
      * k1≤k2→
           Right(CanExplainMe(k1))≤k3→skip;
           ◇Right(CanExplainMe(k1))>k3→
             Best:=Left(CanExplainMe(k1));
             k3:=Right(CanExplainMe(k1));

           k1:=k1+1;

     Status:=In;
     Q_Best!Status;
     k1:=1;
     Status:=Nil;
     * k1≠k2→
         x:=Left(CanExplainMe(k1));
            x=Best→skip;
            ◇x≠Best→
               Q_x!Status;

            k1:=k1+1;
```

Acknowledgments

The research reported in this paper has benefited from discussions with John Josephson, B. Chandrasekaran, V. Krothapalli, Sridhar Vasudewan, and Tom

Bylander. We are deeply grateful to them. We are grateful also to the Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010. and the Air Force Office of Scientific Research. grant 87-0090. for their support.

References

Allemang *et al.*, 1987 Dean Allemang, Michael Tanner. Tom Bylander. and John Josephson. "On the Computational Complexity of Hypothesis Assembly". In the *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. pages 1112-1117, Milan. Italy, August, 1987.

Bylander *et al.*, 1987 Tom Bylander. Dean Allemang, Michael Tanner, and John Josephson. "Some Results Concerning the Complexity of Abduction". Technical Report. Laboratory for Artificial Intelligence Research. Department of Computer and Information Science. The Ohio State University. September. 1987.

Chandrasekaran, 1986 B. Chandrasekaran. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design". IEEE Expert, 1(3):23-30, Fall 1986.

Chandrasekaran. 1987 B. Chandrasekaran. "Towards a Functional Architecture of Intelligence Based on Generic Information Processing Tasks". In the *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. pages 1183-1192, Milan, Italy, August, 1987.

Charniak and McDermott Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, Massachusetts, 1985.

Dijkstra. 1975 Edsger W. Dijkstra. "Guarded Commands, Nondeterminacy and Formal Derivation of Programs". Communications of the ACM, 18(8):453-457. August. 1975.

Goel *et al.*, 1987 Ashok Goel, John Josephson, and P. Sadayappan. "Concurrency in Abductive Reasoning". In the *Proceedings of the DARPA Workshop on Knowledge-based Systems*, pages 86-92. St. Louis. Missouri. April. 1987.

Gomez and Chandrasekaran. 1984 Fernando Gomez and B. Chandrasekaran. "Knowledge Organization and Distribution for Medical Diagnosis". William Clancey and Edward Shortliffe: editors. *Readings in Medical Artificial Intelligence: The First Decade*. Chapter 13, pages 320-339. Addison-Wesley. Reading, Massachusetts, 1984.

Hoare. 1978 "Communicating Sequential Processes" C. A. R. Hoare. Communications of the ACM, 21(8):666-677, August, 1978.

Josephson *et al.*, 1987 John Josephson. B. Chandrasekaran , Jack Smith. and Michael Tanner. "A Mechanism for Forming Composite Explanatory

Hypotheses". IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Causal and Diagnostic Reasoning, SMC-17(3):445-454, May/June, 1987.

Pearl, 1987 Judea Pearl. "Distributed Revision of Composite Beliefs". To appear in the *Journal of Artificial Intelligence*, 1987.

Pople, 1977 Harry Pople. "The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning". In the *Proceedings of the Fifth International Joint Conference in Artificial Intelligence*, pages 1030-1037, Cambridge, MA, August 1977.

Reggia, 1983 James Reggia. "Diagnostic Expert Systems Based on a Set Covering Model". International Journal of Man-Machine Studies, 19:437-460, November, 1983.

Smith *et al.*, 1985 Jack Smith, John Svirbely, Charles Evans, Pat Strohm, John Josephson, and Michael Tanner. "RED: A Red-Cell Antibody Identification Expert Module". Journal of Medical Systems, 9(3):125-138, 1985.

Sticklen, 1987 Jon Sticklen. *MDX2: An Integrated Medical Diagnostic System*. PhD. Dissertation, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, Spring 1987.

Extended Abstract

Distributed Synthesis of Composite Hypotheses

ASHOK GOEL
P. SADAYAPPAN
JOHN R. JOSEPHSON

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

February, 1988

Distributed Synthesis of Composite Hypotheses

ASHOK GOEL, P. SADAYAPPAN, JOHN R. JOSEPHSON

Department of Computer and Information Science
The Ohio State University

Extended Abstract

1. Introduction

A very general problem in Artificial Intelligence is that of synthesizing situation-specific composite structures from stored, more general representations. Design of a device that performs a specific function from available general purpose components is one instance of this problem. Abduction, which typically involves assembling a composite hypothesis that explains an entire data from component hypotheses that can account for portions of the data, is another instance of the same problem. An even more general instance of the problem is formation of schemas. In this paper we propose a distributed mechanism for the general problem of synthesizing composite structures, using abduction as a concrete example to motivate the discussion.

2. Characterization of the Task

The general abductive task is to infer a hypothesis that best explains a set of data [Josephson *et al.*, 1987]. Abduction occurs, for instance, in diagnostic problem solving, where the data is in the form of manifestations (or symptoms), and the explanatory hypotheses are component malfunctions (or diseases). For simple abductive problems, for example, diagnosis under the single fault assumption, a single hypothesis may be sufficient for explaining the entire data, and the abductive task is to find that hypothesis. In general, however, a composite explanatory hypothesis has to be synthesized from component hypotheses each of which can account for some portion of the data.

Let us suppose that we have a set of data D and a set of hypotheses H such that the explanatory coverage $e(h)$ for each h H contains some members of D. Let us also assume that we have a classification scheme that matches each h with D and determines its *prima facie* belief value $b(h)$ depending on the degree of match. Then we may characterize the abductive task as synthesizing a composite hypothesis C that best explains D, where C is a "best" explanation of D if (i) C is complete, *i.e.* $e(C)=D$, (ii) C is parsimonious, *i.e.* no proper subset of C is complete, and (iii) each h C has the highest belief value for explaining some d D. Synthesizing C along these specifications is an instance of the combinatorial optimization problem [Goel *et al.*, 1988]. The problem is underdetermined in that there may exist more than one globally "best" explanation. Further, the problem is non-linear as well as non-monotonic; it is non-linear if two hypotheses in H are incompatible with each other, and it is non-monotonic if two hypotheses in H cancel each other's explanatory capability with respect to some datum in D. Not surprisingly, the general abductive problem has been shown to be *NP-Hard* [Bylander *et al.*, 1988].

We note the correspondence between the synthesis of composite explanatory hypotheses in abduction and design of a device. Indeed, if we view the composite hypothesis as an abstract device whose function is to explain some data then the problems of abduction and design are equivalent. [Goel *et al.*, 1988]. Thus the requirement of complete explanatory coverage of data is the goal of designing a composite hypothesis, and inclusion of hypotheses with maximal belief values are the subgoals. The incomptability and cancellation interactions impose local constraints on the choice of explanatory hypotheses for accomplishing these goals, while the requirement for a parsimonious composite hypotheses represents a global constraint. A corollary of this equivalence between the abduction and design problems is that the general design problem is *NP-Hard* as well.

3. Multiple Perspectives

The general mechanism that has been used for synthesizing composite structures is the generate and test method. In the case of synthesizing composite explanatory hypotheses [Goel *et al.*, 1987a; 1987b; 1988], the generation phase produces a composite hypothesis that (i) satisfies the requirement of complete explanatory coverage, (ii) includes component hypotheses with maximal belief values, and (iii) accommodates interactions between the components. In the test phase, the generated composite hypothesis is tested for parsimony, and improved if possible. While the generate and test method has been successfully in the construction of knowledge-based systems for simple domains, it is a "weak" method with some inherently sequential aspects to it.

The power of this method can be enhanced and the implicit concurrency in it exploited by adopting *multiple perspectives*. In synthesizing composite explanatory hypotheses, for instance, there are two distinct perspectives. From the perspective of hypotheses, each hypothesis *h* asks "which elements of *D* can I be used to explain?". This question can be answered for each *h* *H* concurrently with others. Similarly, from the perspective of data, each datum *d* asks "which hypothesis can best explain me?". Again, this question can be answered for each *d* *D* concurrently with others. If we associate a process with each *h* *H* and each *d* *D* then the control of information processing continuously shifts from the hypotheses processes to the data processes, and *vice versa* until a composite hypothesis *C* that best explains *D* has been synthesized. Communication between the processes is achieved by passing semantically encoded messages. Thus an ensemble of semi-autonomous agents views the same problem from different perspectives and cooperatively arrives at a solution. In the full paper we show just how a composite hypothesis can be synthesized in this fashion.

4. Conflicts, Negotiation, and Intervention

In the mechanism that we have outlined above, the explanatory hypotheses compete with one another for inclusion in the composite hypothesis. This leads to conflicts between them since each competing hypothesis has access to only its own local view of the global problem. An instance of this conflict occurs in the testing of a composite hypothesis for parsimony where explanatory superfluous hypotheses are removed from the composite. A similar conflict arises in dealing with the incompatibility interactions between the hypotheses.

The general conflict resolution strategy that we adopt is that of *negotiation* between hypotheses with conflicting interests. The competing hypotheses negotiate with one another when a conflict between them arises by exchanging messages, and resolve the conflict on the basis of their belief values. However, under certain conditions negotiations may fail to resolve the conflict. An example of this is when negotiations between the hypotheses are deadlocked due to formation of cycles in the negotiation process. In such situations *intervention* by some higher process is required. Our model provides for such interventions in order to break the deadlock in negotiations. We note that implicit in the intervention process is the notion of hierarchicalization of the synthetic process [Goel *et. al.*, 1987a].

5. Conclusions

We have developed a model for distributed synthesis of composite structures from stored, more general representations, and illustrated it for the specific problem of abductive explanation. The model can be implemented on a distributed memory, message passing, parallel computer architecture such as the Hypercube machine [Goel *et. al.*, 1988]. However, the model itself is at the level of information processing tasks, behaviors, and abstractions. The model involves multiple perspectives, and uses the conflict resolution strategies of negotiation and intervention when needed.

An interesting variation on the problem is that of abductive explanation by a collective of agents. In the domain of medical diagnosis, for instance, the clinical physician diagnosing a patient case may rely on a pathologist for explaining biopsy data and on a radiologist for explaining x-ray data. The model that we have described can be extended to accommodate such collective synthesis of composite explanatory hypotheses.

Acknowledgments

The research reported in this paper has benefited from discussions with B. Chandrasekaran to whom we are deeply grateful. We are grateful also for the support of the Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010, and the Air Force Office of Scientific Research, grant 87-0090.

References

[Bylander *et al.*, 1987] Tom Bylander, Dean Allemang, Michael Tanner, and John Josephson. "Some Results Concerning the Complexity of Abduction". To be presented at the *AAAI Symposium on Artificial Intelligence in Medicine*, Stanford University, Palo Alto, California, March 22-24, 1988.

[Goel *et al.*, 1987a] Ashok Goel, John Josephson, and P. Sadayappan. "Concurrency in Abductive Reasoning". In the *Proceedings of the DARPA Workshop on Knowledge-based Systems*, pages 86-92, St. Louis, April, 1987.

[Goel *et al.*, 1987b] Ashok Goel, P. Sadayappan, John Josephson, and N. Soundarajan. "Distributed Synthesis of Composite Explanatory Hypotheses". Technical Report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State

University, November, 1987.

[Goel *et al.*, 1988] Ashok Goel, P. Sadayappan, and John Josephson. "Concurrency in Design of Composite Explanatory Hypotheses". Technical Report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, January, 1988.

[Josephson *et al.*, 1987] John Josephson, B. Chandrasekaran , Jack Smith, and Michael Tanner. "A Mechanism for Forming Composite Explanatory Hypotheses". *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):445-454, May/June, 1987.