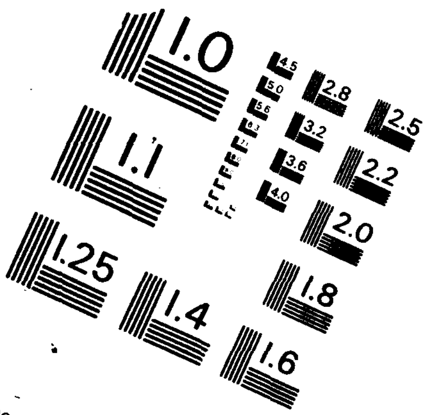
AD-A170 830 DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS B CHANDRASEKARAN SEP 85 1/1

UNCLASSIFIED AFOSR-TR-86-0509 AFOSR-82-0255 F/G 9/4 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

RF Project 763180/714659
Annual Report

DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL

B. Chandrasekaran
Department of Computer & Information Science

For the Period
July 1984 - June 1985

DEPARTMENT OF THE AIR FORCE
Air Force Office of Scientific Research/PKZ
Building 410
Bolling Air Force Base, D.C. 20332

Grant No. AFSOR-82-0255

August 1985

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

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 86-0509	2. GOVT ACCESSION NO. ADA170 830	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DISTRIBUTED KNOWLEDGE BASE SYSTEMS FOR DIAGNOSIS AND INFORMATION RETRIEVAL		5. TYPE OF REPORT & PERIOD COVERED Annual Report 7/1/84 - 6/30/85
		6. PERFORMING ORG. REPORT NUMBER 763180/714659
7. AUTHOR(s) B. Chandrasekaran		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0255
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Ohio State University Research Foundation, 1314 Kinnear Road Columbus, Ohio 43212		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304-K1
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Air Force Air Force Office of Scientific Research Bolling Air Force Base, D.C. 20332		12. REPORT DATE September 1985
		13. NUMBER OF PAGES 98
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Expert systems, diagnostic reasoning, deep models, functional representations, expert systems for design, qualitative simulation, the CSRL language, consolidation, generic tasks in expert systems, knowledge acquisition, explanation

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

>During the year, progress was made in our research on distributed approaches to knowledge-based problem-solving in the following areas: 1) We have developed an approach called consolidation to reason qualitatively about the behavior of systems of components. This approach and the more classical approach of qualitative simulation are complementary. 2) We have made further progress in developing an architectural framework for diagnostic reasoning. 3) We have elucidated some of the criteria that govern how

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Block 20 (Abstract) - Continued

design plans are selected for further refinement in design problem solving.
4) We have identified a number of generic tasks into which the information processing activity of most of the expert systems can be decomposed. These generic tasks are at a much higher level of abstraction, and this should make knowledge acquisition and explanation for expert systems easier.

5) We have clarified how symbolic qualitative knowledge-based processing helps when problems get complex by considering the concrete task of classification. We compare the pattern recognition and AI approaches to the problem.

RF Project 763180/714659
Annual Report

Distributed Knowledge Based Systems For Diagnosis and Information Retrieval

B. Chandrasekaran
Department of Computer and Information Science

For the Period
July 1, 1984 - June 30 1985

DEPARTMENT OF THE AIR FORCE
Air Force Office of Scientific Research
Bolling Air Force Base, D.C. 20332

Grant No. AFOSR-82-0255

August, 1985

1. Technical Progress During Period July 1, 1984 - June 30, 1985

During the year covered by this report, technical progress was made in a number of directions in our project, "Distributed Knowledge Base Systems for Diagnosis and Information Retrieval." Our recent emphasis on the general area of *reasoning about complex devices and systems* has continued. The ubiquity of complex devices in our technological life and the importance of keeping them operational is a sufficient justification for studying how human experts reason about them and how such reasoning can be incorporated in computer systems. From a purely scientific viewpoint, these forms of expert reasoning provide a good experimental arena for theories of knowledge representation and problem solving.

Our long-term goals are to understand the structure of knowledge and the reasoning processes needed for producing computer-based expert decision-making and consultation systems. We would like our systems to have a measure of *understanding* of the domain, different types of problem solving strategies, and have a conceptual structure that matches that of humans in the same domain, so as to facilitate knowledge acquisition and user interaction. Our near-term goals are to pursue this aim in the specific task domains of *diagnostic reasoning* and *automated design.*

During the year under report, we especially concentrated on:

1. *diagnostic reasoning* about complex systems;
2. *automation of expert design behavior of certain types*;
3. giving diagnostic expert systems a measure of *understanding* of their domain; and,
4. laying the conceptual foundations of an approach to expert system design at a much higher level of abstraction than is currently common.

Basic research on *diagnostic reasoning* includes:

1. Control strategies and knowledge organization. *Issues*: which diagnostic hypothesis to consider when? How should diagnostic knowledge be distributed among various hypotheses?
2. Causal models (we call them "functional models"). They describe how devices actually work. Aim: to be able to automatically generate diagnostic hypotheses and diagnostic knowledge from these models.
3. Investigation of how to reason about the *qualitative behavior* of complex systems, much like human experts do. Traditional approach of qualitative simulation is useful where applicable, but when a large number of components involved it may result in *combinatorial problems.* Our approach: *consolidation* of behavioral abstractions of components into a whole without the need for actual simulation. Goal: to be able to give AI systems the ability to put together descriptions of how systems of

components will behave, given behavioral descriptions of components themselves.

Basic research on design problem solving includes:

– understanding the different kinds of design knowledge and strategies used by human experts.

–we have categorized design into 3 classes, in terms of complexity and types of knowledge and problem solving. We have shown how to automate Class 3 design, which corresponds to what one might call "routine" design by expert designers.

2. Project Progress Reports

Following the tradition of earlier years, we will present the specific progress during the year by summarizing the contents of *new* papers that were prepared, submitted or published during the year. The papers themselves will be attached as appendices to this report. A number of papers that had been submitted as appendices to the previous annual reports were published during the year, but they will not be included in this report.

1. *Qualitative and Functional Reasoning about Physical Systems and Devices.* The "Consolidation" Technique for Reasoning about Behaviors of Systems of Components: Commonsense reasoning includes, among others, an ability to efficiently derive qualitative accounts of systems of physical components behave. AI research of the past few years has paid increasing attention to this problem. A class of approaches called *qualitative simulation* has been proposed for this class of problems. Essentially, each component is described in terms of qualitative changes its important variables undergo in response qualitative changes in some of its input variables, and a calculus by which these changes can be propagated through the components. The progress in our research has been in two parts – one, in understanding the limitations of qualitative simulation, which has so far been the main method that has been available for inferring the behaviors of systems from the behaviors of the components; and two, in understanding the prospects for an alternative method. In Appendix A, "A Critique of Qualitative Simulation from a Consolidation Viewpoint," we compare two main techniques for qualitative simulation with the consolidation approach. In Appendix B, "Understanding Behavior Using Consolidation," we give details of the consolidation approach. Appendix C describes both our functional representation and the consolidation investigations in the context of diagnosis.

2. *Diagnostic Reasoning.* As we have indicated in earlier reports, we have been developing a family of higher level languages for expert system design, with partial support from this grant. The CSRL language is, particularly useful for diagnostic system design, and we have reported on

it widely in the literature. Appendix D, "Mapping Medical Knowledge into Conceptual Structures," describes how a complex body of knowledge may be analyzed properly so that it can be encoded adequately in this language. The research for the paper was partially supported by the AFOSR grant.

Also in the area of diagnostic reasoning, Prof. Chandrasekaran co-edited a special issue of the SIGART Newsletter on "Structure and Function in Diagnostic Reasoning." We include as Appendix E the editorial of this special issue, which gives an architecture for diagnostic reasoning. This paper organizes the research in this area into a framework that we think may be useful in understanding the difference in motivations between different research activities in the area.

3. *Design Problem Solving.* The year saw the completion of the Ph. D dissertation of David C. Brown, *"Expert Systems for Design Problem Solving Using Design Refinement with Plan Selection and Redesign."* This research was supported by the AFOSR grant. We have forwarded copies of the dissertation to AFOSR earlier, and in the last year's annual report had included an extensive summary of the research. Appendix F, "Plan Selection in Design Problem-Solving," describes the specific technical issues in how design plans are selected.

4. *Theoretical foundations of knowledge-based reasoning.* For a number of years we have been arguing that much of the discussion in knowledge-based reasoning has been at *toc low a level of abstraction*, and that phenomena at the *knowledge level* exist and need to be explored. Essentially the idea is that the current emphasis on rules, frames or logical languages is really an emphasis on *implementation-level* issues. A useful level of abstraction is one which considers different types of knowledge and different types control regimes for different kinds of generic tasks. The idea is that if we have a repertoire of such generic tasks with associated characterization of types of knowledge, then complex knowledge-based reasoning tasks can often be decomposed into an interacting collection of modules, each of which performs one of the generic tasks. In our research we have developed an open-ended repertoire of such tasks and show how most of the expert systems extant can be viewed as combinations of the generic tasks that we have identified. As Appendix G we enclose "Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving," which describes these ideas.

   One of the fundamental paradigmatic characteristics of AI is the fact that it uses complex *symbolic and qualitative structures* for representation and manipulation as opposed to numerical information. The task of classification is one of the tasks for which AI expert systems have been most successfully deployed: systems such as Mycin, MDX and Prospector can be viewed as performing some version of the classification task. Now, for more than two decades there has been a field called pattern recognition, which has used ***statistical classification*** as a basic tool. We have compared the numerical approaches of pattern recognition to the symbolic approaches of AI to the same classification task in an attempt to learn where and why symbolic and qualitative structures that so characterize human thought emerge. Results of this investigation appear as Appendix H, "From Numbers to Symbols to Knowledge Structures: Pattern Recognition and Artificial Intelligence Perspectives on the Classification Task."

3. Publications Prepared During the Year on Research Supported by the Grant

1. B. Chandrasekaran, co-editor of *ACM SIGART News* Special Issue on "Structure, Behavior and Function in Diagnostic Reasoning," July 1985. Editorial, with Robert Milne,with the above title. (Appendix E.)

2. B. Chandrasekaran, Tom Bylander and V. Sembugamoorthy, *Functional Representation and Behavior Composition by Consolidation: Two Aspects of Reasoning about Devices,* SIGART Newsletter, Special Issue on Structure, Behavior and Function, No. 93, July 1985, pp. 21-24. (Appendix C.)

3. B. Chandrasekaran, *Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving,* To appear in the Proceedings of the National Academy of Sciences / Office of Naval Research Workshop on AI and Distributed Problem Solving, May 16-17, 1985. (Appendix G.)

4. B. Chandrasekaran, *From Numbers To Symbols To Knowledge Structures: Pattern Recognition and Artificial Intelligence Perspectives on the Classification Task,* Paper presented at the Workshop on Pattern Recognition in Practice-II, June 19-21, 1985, and to appear in the book *Pattern Recognition in Practice-II.* (Appendix H.)

5. Tom Bylander, *A Critique of Qualitative Simulation From a Consolidation Viewpoint,* To appear in the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, November 12-15, 1985. (Appendix A.)

6. B. Chandrasekaran and Tom Bylander, *Understanding Behavior Using Consolidation,* To appear in the Proceedings of the 9th International Conference on Artificial Intelligence, Aug 18-24, 1985. (Appendix B.)

7. Tom Bylander and Jack W. Smith, Jr., M.D., *Mapping Medical Knowledge Into Conceptual Structures,* To appear in the the Proceedings of The Expert Systems in Government Symposium, October 24-25, 1985. (Appendix D.)

8. B. Chandrasekaran and David C. Brown, *Plan Selection in Design Problem-Solving,* Appears in the Proceedings of The Society for AI and Simulation of Behavior 1985 Conference, April, 1985. (Appendix C.)

WELCH

COVER-PAGE.X27

OSU-2Ø

```
Job:  COVER-PAGE.X27
Date: Fri Sep 27 09:25:54 1985
```

Appendix A

Special Section on

Reasoning About Structure, Behavior and Function

B. Chandrasekaran[1] and Rob Milne[2], Guest Editors

Introduction

The last several years' of work in the area of knowledge-based systems has resulted in a deeper understanding of the potentials of the current generation of ideas, but more importantly, also about their limitations and the need for research both in a broader framework as well as in new directions. The following ideas seem to us to be worthy of note in this connection.

* There is increasing interest in the multiplicity of knowledge structures and problem solving techniques that seem to underlie complex reasoning tasks. When viewed at the implementation language level, systems often seem to have relatively simple and uniformly represented pieces of knowledge and control structures, but viewed at the task level, they often do fairly complex kinds of actions. E.g., MYCIN, viewed as rule-based system, has knowledge encoded within the simple rule formalism, and its control structure is again a simple backward chaining. But viewed as a diagnostic problem solving system, it has a much more complex knowledge structure and problem solving regime, all of which happen to be implemented in a rule-based formalism. There is considerable interest in trying to understand the tasks themselves. Thus there is interest [1-4] in the class of problems that can be called diagnostic problems or design problems, or classification problems.
* While rule-based languages are, logically, computation-universal, and thus can incorporate any kind of knowledge, in practice, most of the systems have encoded what have been called associational knowledge, i.e., pieces of knowledge that go from data in the domain to partial conclusions of interest. Thus diagnostic systems would typically have most of their knowledge in the form, "Observations --> diagnostic hypothesis." Often such knowledge may not be available, or may become too large in number in complex domains, or the knowledge base may be incomplete. In these cases, it would be useful to have methods by which the system can use a model of the domain or system under consideration, and reason with this model to generate information about the expected behavior of the system. Such models have been variously called causal or deep models. There has been significant interest in representing and manipulating such models.

[1]Laboratory for AI Research, Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210. His work in this editorial activity was supported by AFOSR grant 82-0255

[2]Army AI Center, HQ DAIM-DO, Pentagon, Washington, DC 20310

In fact these two trends come together rather well in a body of recent work that deals with reasoning about devices, where the general concern is one of how the behavior of a device is related to and arises from its structure. On one hand, the general motivation behind this class of work is to understand the multiplicity of processes that are needed for a complete account of diagnostic reasoning as a task (i. e., concern with architecture of task-specific reasoning as opposed to concern with implementation-level concerns at rule, frame or logic language level). On the other hand, such an understanding of the relation between structure, behavior and function of a device is precisely the causal model for the diagnostic task in that domain: it explains how the behavior (or malfunction) is caused by the structural properties of the device. As we shall see later in some of the papers included in the special issue, they are also deep models in the sense that they correspond in some cases to some degree of understanding of the device, and can be used to derive the more associational pieces of knowledge used by the shallower problem solving systems.

Structure, Behavior and Function: Relation to Diagnostic and Other Reasoning Tasks

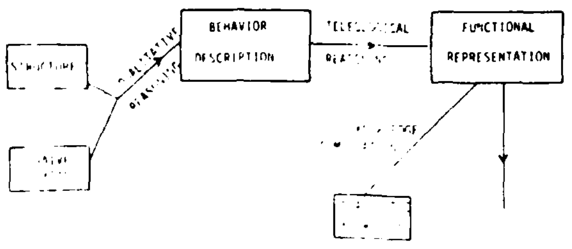
In order to get a better understanding of the different processes and types of knowledge involved in diagnostic reasoning, it will be useful to consider some typical diagnostic scenarios.

Typically, a diagnostic problem starts with the observation of some behavior which is recognized as deviation from the expected or desirable, i.e., a malfunction behavior is observed. The problem solver at this stage needs to generate some hypotheses about the cause of the malfunction: typically these are in terms of changes in the structure of the device from the specifications. In areas such as medicine, at this stage a number of low-cost broad spectrum testing (such as physical examination, a battery of blood tests, etc.) may be undertaken without any specific hypotheses in mind, or the initial malfunction may be used to invoke one or more specific malfunction hypotheses. Most often these hypotheses are invoked by using what one might call "precompiled" pieces of knowledge that relate behavioral observations to one or more hypotheses. This initial hypothesis generation task can be more or less complex, and more or less controlled depending upon the domain, and the knowledge the problem solver has. Whatever the particular method, they all involve going from behavioral observation (test values, signs and symptoms, etc.) to a number of hypotheses, possibly ranked.

At this stage typically a small number of the more plausible hypotheses are considered the differentials. In a compiled system, knowledge may be explicitly available for each hypothesis in the differential about which further tests may be useful for confirmation or rejection of that hypothesis, and in that case by comparing this knowledge for the different hypotheses in the differential, the problem solver can generate tests that have the potential for the greatest discrimination between the hypotheses. If, however, this knowledge is not directly available to the problem solver, but the structure of the device is known then the following reasoning can be very useful. Assume the structure change corresponding to each of the malfunction hypotheses in the differential list, and reason about what behavior will follow. One would like to do this qualitatively in general for a number of reasons (see the paper by Forbus in this collection for these reasons). The basic work in this area was initiated by de Kleer [1], and he and Brown at Xerox PARC, Forbus, and Kuipers (see references in Forbus' paper in this issue) are among those who have continued this line of work. In this special issue, Forbus reports, among others, on work of this type, and gives the motivation of the research, and the computational constraints on the kinds of solutions. Reasoning from a given structure to its behavior is required not only in diagnostic reasoning; it is also useful in design, where the problem solver will need to project a design's behavior to check conformity to design specifications, and in planning, for very similar reasons.

Architecture of Causal Reasoning

Causal reasoning about devices or physical systems involves multiple types of knowledge structures and reasoning mechanisms. The following diagram schematically indicates some of the components of this. In what follows, we will attempt to relate the papers in the special issue to the issues as we identify them.



For purposes of our current discussion, the following stages can be recognized in causal reasoning.

1. Given a representation of the behavior of the components of a device or system and a representation of the structure of the device, i.e. the interconnection of the components, the ability to generate the behavioral description of the device as a whole is an important part of causal reasoning. In simple devices or systems this stage will generate enough information to understand the device. But in general this technique is useful for producing various fragments of behavior for ranges of values of components. Often these fragments may need to be further organized to explicitly represent the hierarchical structure of the device and also to capture the teleology of the device as in 2 below.

It is to be noted that as a rule in addition to behavioral descriptions of components substantial amounts of domain knowledge or general common sense knowledge may be needed for this reasoning. In the diagram this is indicated by the box "naive physics" knowledge. But in specific domains instead of naive physics knowledge domain-specific knowledge (such as various laws of electricity) will be needed.

In this special issue Forbus presents, among others, an account of how a qualitative account of behavior can be obtained given a structural description of objects and their connectivity. Bylander and Chandrasekaran in their work on consolidation reported here discuss how the behavior of composites of components may be put together from the behavioral descriptions of the components.

Cross describes how qualitative reasoning from structure to behavior is useful in evaluating proposed plans in the air traffic control domain. Both Cross and Forbus are also concerned with how qualitative reasoning and quantitative models may be combined in certain situations. Stanfill looks at some naive physics aspects of reasoning about simple machines. Relationship between the spatial properties of the components and possible motions is an important part of this work.

White and Frederiksen discuss qualitative reasoning in the domain of circuits. The context of the work is in teaching trouble-shooting (as opposed to automated diagnosis). The work is conducted in the domain of automotive electrical systems.

2. Given the ability to generate behavioral sequences for various assumptions about the components, the agent can often put together an account of the functions of the device, and its relationship to its structure. In simple cases, the behavior that we talked about in 1 above can be the function, but in general, functional specifications involve teleology, i. e., an account of the intentions for which the device is used. Also often behavior may need to be abstracted to a level higher than that at which the component is specified. E. g., in an electronic circuit, the behavior of the components such as a transistor and a resistor may be in terms of voltages and currents, while a device containing them may be described as an amplifier or oscillator. To go from the level of description in terms of "currents" and "voltages" to one of "amplification" and "oscillation" requires an abstraction process. This abstraction process often involves a hierarchical organization of representation of the relation between function and structure.

Sembugamoorthy and Chandrasekaran discuss the nature of such a functional representation, and propose that it captures in some sense an agent's understanding of how the device functions. In general how an agent constructs a functional account from the structure and behavioral specifications of the components is an interesting theoretical question. de Kleer [5] has provided some examples of this process.

Simmons, in his paper in the special issue, outlines the issues in representing in a graphical (and animated) form the functioning of devices and relating this to natural language descriptions of how the device works.

3. While the stages so far help in understanding how the device works, these structures will need to be used in specific ways to help in specific problem solving tasks. The most commonly studied such task is diagnostic reasoning. Often, one can generate diagnostic possibilities (malfunction modes) and test data that will help in determining the presence of these from one's understanding of how the device works, or in specific domains by knowledge about the components and their behaviors and functions. The paper by Sembugamoorthy and Chandrasekaran outlines how their functional representation can be manipulated by device-independent processes to produce diagnostic knowledge of this type.

As another example, in the HELIOS system described by Kramer [6] the behavior of the devices and their structure are used to propose possible faults and generate tests in order to confirm these faults. This system uses the design description of digital circuits to both confirm the design and diagnose faults in the system. Expected behaviors can be propagated through the structure in order to diagnose faults. Conflicts in this propagation are used to generate diagnostic possibilities and tests are then proposed to confirm or deny the candidates.

In this Special Section, Davis, et al, report on what has already received wide attention: their work on doing diagnosis of digital circuits by using knowledge of structure and function. Because of the digital nature of the outputs, the distinction between behavior and function is not as clearcut as in the work of Sembugamoorthy and Chandrasekaran, but both works emphasize the hierarchical nature of the functional representation.

Milne presents an approach where portions of the intended behavior of the device can be directly traced to certain components, i. e., the component is responsible for that part of the output behavior. Of course, in general there will not be such simple mappings between functions and components, but whenever such a mapping is possible, diagnostic knowledge can be easily generated relating undesired behavior to component malfunctions. Cantone, et al, describe work which has similarities to Milne's work. In their work, a data base of relationships between generic components and the kinds of device behavior they contribute to is assumed available, and as the structure of a particular device containing these components is given, their approach provides a means in some cases of automatically putting together a collection of diagnostic relations between observations and malfunctions.

Use of such causal models for other tasks than diagnosis is of interest. Cross uses qualitative simulation for evaluating plans, and White and Frederiksen use it for generating explanations of device functioning in teaching. Beck and Prietula also propose using it in teaching pathophysiology to medical students. Milne uses qualitative simulation in order to derive the responsibility each component has in the final output. The representation in Sembugamoorthy and Chandrasekaran can be used for certain classes of "What will happen if..." questions about a device.

Long describes a structural representation and some processes that operate on it for reasoning about the pathophysiology of heart diseases. The use of causal models in this paper is rather unusual: as he points out it is a sort of "visi-calc" for the patient data base. As new data are entered, the information stored in the structural model of the pathophysiology is used to check consistency and make projections of values of other relevant data items.

Hudlicka and Lesser take the novel approach debugging problem solving systems by modeling their structure and function in analogy with physical devices.

Diagnostic Strategies

In this section, let us elaborate on the origin and use of what one might call structure and function expert systems.

In the beginning was the rule-based expert system. Knowledge was represented in the form of production rules. The inference engine was a very simple mechanism. It merely found the set of rules that could conclude the current hypothesis, and tried each one in turn. On occasion, in order to satisfy a rule, other rules would be needed. This lead to recursion and backward chaining.

This approach worked for simple problems. In the early days of expert systems, the field could only do simple problems. Expert systems could be developed because the inference engine gave a very simple paradigm for solving problems. After a time, however, problems became too complex for simple rule bases. Especially in diagnostic reasoning, as mentioned earlier, structure and function have begun to be used to guide the effort to isolate the fault. Function is used two ways: first to help isolate the fault in the structure, and then to further reason about the possible fault.

The simplest use of structure is outlined in Milne's paper. By intersecting paths known to be good and bad, faults in some portions of the system can be ruled out. This decision is based solely on GO/NOGO information. The simplest algorithm is to split the possible fault path in half. In order to pick the optimal test to perform next, information about the cost of each test and the possible value of the test can be considered. Cantone has one of the most elaborate algorithms for this task.

Eventually, the presence or absence of faults becomes insufficient to reduce the number of possible candidates any further. At this time, the expected outputs of the devices which make up the system can be used to further identify the faults. By examining how the function of each device should alter its inputs, candidates can be eliminated. Davis uses the propagation of values and the function of digital devices to rule out possible faults. Genesereth [7] further reduces the possible faults by reasoning which inputs could not produce the wrong output. For example, if an AND gate has a 0 on input1 and a 1 on input2 and produces 1 on the output, then we conclude that input1 is at fault since its 0 value was responsible to cause the output to be 0. This approach reduces the search considerably.

Scarl, et al., in this Special Issue describe a more complicated reasoning mechanism to declare the innocence of devices by inferring that their function could not have caused the possible fault. This work illustrates the reasoning and issues involved in using function in this way. White, et al. use a simple view of function in electrical circuits to rapidly guide a binary search for the fault. Their approach is at the other end of the spectrum from Scarl's work. Whereas Scarl uses a complex combination of reasoning, White's simple method is just as effective, although not as general.

If there is no output at all, then no information is present on which to base reasoning about the function. In this case Milne uses the ways devices can behave such that they produce no output and propose faults. For example no current will flow through a resistor if it is open. In a typical series circuit, there is only one path for the current, so each resistor could be open. This is analogous to the Christmas tree light problem.

When the structure cannot be used to further isolate the fault, then function alone must be used. Often in electronic circuits, it is not possible to test between groups of components. In this function can be used to deduce the possible fault. The work of Davis and

Appendix B

Functional Representations and Behavior Composition by Consolidation: Two Aspects of Reasoning about Devices[5]

B. Chandrasekaran, Tom Bylander, and V. Sembugamoorthy[6]
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210 USA

In our laboratory, we have a number of projects on diagnostic reasoning in both medical and mechanical domains. Our work on the MDX system [2] considered medical diagnosis as largely a classification problem solving activity, and viewed diagnostic knowledge as a collection of precompiled associations between manifestations and classificatory diagnostic hypotheses. More recently, we have extended our investigations in a number of directions, two of which are relevant to the purpose of this special issue.

1. One way that an agent may generate diagnostic knowledge is by deriving it from an understanding of how the device or system under consideration works. Here the concerns are: What is the nature of the representation in which the agent's understanding is encoded? What are the device-independent processes which can operate on the representation to produce diagnostically useful knowledge? Finally, what other kinds of problem solving, other than diagnosis, can be supported by a functional representation?
2. Given a description of possible behaviors by a component and given a collection of components connected in a certain way, how can the component behaviors be composed into a behavioral description of the collection as a whole? The above question is motivated by what we felt was a need to seek alternatives to qualitative simulation, which has been the most common approach to generate behaviors from component descriptions.

In the following sections, we outline our research in these two areas.

Functional Representation of Devices as Deep Models

V. Sembugamoorthy and B. Chandrasekaran

Human experts often use in their problem solving a deeper understanding of their knowledge domain than has been captured in the first generation of expert systems. Several aspects of this deeper understanding are being investigated under the terms causal reasoning and qualitative physics in both medical and non-medical domains [6, 3, 5]. We have been working on the aspect of functional representation, which is an expert's understanding of how the functioning of a complex device results from its structural properties. Such a representation can then be operated upon by device-independent compilers for producing problem solving structures of various kinds. We have actually built such a compiler which automatically generates a diagnostic expert system from the functional representation of a device. In this report, we outline the main ideas. The details are available in Sembugamoorthy and Chandrasekaran [7].

The idea is that an agent's understanding of how a device works is organized as a representation that shows how an intended function is accomplished as series of behavioral states of the device, and how each behavior state transition can be understood as either due to a function of *a component, or in terms of further details of behavior* states. This can be repeated at several levels so that ultimately all of the functions of a device can be related to its structure and the functionality of the components in the structure. For example, the function that we may call "buzz" of a household electric buzzer (an example system used by de Kleer and Brown) may be represented as:

```
FUNCTION: Buzz : TOMAKE buzzing(buzzer)
          IF pressed (switch)*
          by behavior1
```

and the relevant behavior, behavior1, can be represented as in figure 1.

```
BEHAVIOR: behavior1:

 Pressed(switch)*
      |
      | BY behavior2
      V
{Clapper electrical connection alternates}
      |
      | USING-FUNCTION mechanical OF clapper
      V
Repeated-Hit(Clapper)
      |
      | USING-FUNCTION acoustical OF clapper
      V
Buzzing(Clapper)
      |||
      |||
Buzzing(Buzzer)
```

Figure 1. Behavior1 of the buzzer

Intuitively what is being said is that the Buzz function is accomplished when, if the switch is pressed, the buzzer goes to a state called buzzing, and this is accomplished by a series of behavioral states that is named behavior1. Behavior1 says that the buzzer, on the occasion of the switch being pressed, goes to a state where the electrical connections in the clapper alternately close and open, which results in the state where the clapper is repeatedly hit, which results in the buzzer being in the state of buzzing. Each transition is further explained, either in terms of further details in the state transition, or in terms of the functions of the components. For example, the transition from the clapper being alternately electrically connected and disconnected, to its being in the repeated-hit state, is explained by relating it to the mechanical function of the clapper.

[5]This research is supported by National Science Foundation grant MCS-8305032 and Air Force Office of Scientific Research grant AFOSR 82-0255

[6]V. Sembugamoorthy is currently at Schlumberger Well Service, P.O. Box 200015, Austin, Texas 78720 USA

Let us see how this fragment of functional representation can be used to generate a piece of diagnostic knowledge that may be used by a diagnostic expert system. A diagnostic compiler will function as follows. Suppose a buzzer does not buzz when its switch is pressed. In order to find out what malfunctions are causing this, the diagnostic compiler will reason thus on the basis of the functional specification and the behavior1 specification: The functional specification tells it that the problem is in behavior1, since the Buzz function is failing. Behavior1, on examination, can result in a series of hypotheses, e.g.:

* If switch is pressed, but the clapper is not alternately electrically connected and disconnected, problem is in behavior2.
* If switch is pressed, the clapper's electrical connectivity alternates, but the clapper doesn't hit-repeatedly, the cause of buzzer not buzzing is some mechanical malfunction of the clapper.

The power of this method for representing how a device works is due in large measure to explicitly distinguishing five aspects of an agent's understanding of the device, and treating each aspect appropriately. The distinctions hold at every level of organization on which the device is represented. The five aspects are:

* **STRUCTURE** - this specifies the relationships between components.
* **FUNCTION** - this captures the intended purpose of a device or component, specified as what the response is to a stimulus.
* **BEHAVIOR** - this specifies how, given a stimulus, the response is accomplished.
* **GENERIC KNOWLEDGE** - chunks of deeper causal knowledge that have been compiled from various domains to enable the specification of behavior.
* **ASSUMPTIONS** - other specifications of the conditions under which various behaviors or conditions occur.

In our research we have identified three dimensions to a functional representation: causal, which accounts for how function which arise from causal chains can be represented; temporal, which represents the temporal relationships within and between each causal event; and what we have called the communication dimension, which accounts for information exchange from different subsystems. So far we have only developed the representational language for the causal dimension, and we are currently working on extensions to other dimensions.

A "deep" model, such as the one outlined above, will be particularly persuasive if can support more than one type of problem solving activity. We have briefly indicated its usefulness in supporting diagnostic reasoning. It can also be used to support a form of predictive reasoning of the type: "What will happen if < >?" For example, if one were to ask, in the buzzer case, "What will happen if the clapper is malfunctioning acoustically?," it is easy to identify behavior1 as the one that will be affected, in particular to infer that while the clapper arm will continue to hit the clapper repeatedly, it will fail to make the buzzer buzz.

Directions for future research include the following: We need to develop methods to check the correctness/consistency of a given device representation. We need to investigate the design of the two other dimensions of device representation. Also the causal dimension has to be integrated with the other two in a disciplined, practically useful, and cognitively meaningful framework. We need to identify the compilation processes that come into play to generate other types of expert problem solving structures, such as the predictive reasoning just discussed.

In broader terms, this work is part of our on-going effort to uncover the multiplicity of generic structures and processes involved in knowledge-based problem solving. Whether or not one accepts the hypothesis that homogeneous and unitary architectures such as production systems are adequate at the level of symbol processing in the mind, we nevertheless believe that in order to account for knowledge-based problem solving activity at the information processing level there is a need to identify a richer collection of generic knowledge structures and a correspondingly rich collection of knowledge-processing mechanisms that operate on them.

Qualitative Reasoning About Physical Systems

Tom Bylander and B. Chandrasekaran

A recent AI approach for reasoning about the behavior of physical systems is qualitative simulation. The structure of the physical system, and knowledge about the behavior of its components are used to derive a collection of constraints. Using these constraints, the simulation is performed and its results are interpreted. This research investigates a new method of reasoning for this problem which we call consolidation. Again, only the main ideas will be described here. Further discussion can be found in Bylander and Chandrasekaran [1].

The usefulness of investigating this form of reasoning can be seen within the context of diagnostic reasoning, as follows. Often an agent does not have direct diagnostic knowledge about a system. In that case, he has to resort to the deep knowledge structures both about the device and about the physical world in order to answer question of the form: "What behavior will follow if a particular structural change (a malfunction) took place?" In the absence of compiled knowledge for this task, the agent will need to put together, i.e., derive, a behavior from his knowledge of the behavior of the components.

The major processing sequence of consolidation is to hypothesize a composite component consisting of a selected subset of components, and then to infer the behavior of the composite from the behaviors of the components. Successful application of this sequence on increasingly larger composite components results in inferring the behavior of the whole system. As a byproduct, a hierarchical behavior structure is produced which explains how the overall behavior is caused by the components' behavior. Also note that each reasoning step is localized over a small number of components and subsystems, avoiding the global problem solving required for qualitative simulation.

This research also proposes a novel representation for behavior. Current theories describe behavior as constraints and operations on the components' quantities and derivatives of quantities, which would imply that consolidation is mainly a matter of algebraic manipulation. Instead, we describe the behavior of a component by the actions that the component performs upon "substances," e.g., fluids, electric currents, control activations, or other stuff that can potentially move. We claim that there is a small set of behavior schemas which can directly represent these actions, and which allow inferences about the behavior of composite components. It is this inferential capability which gives consolidation credibility since otherwise, complex algebraic problem solving is required. Some of the schemas which we have identified so far are:

1. **Allow**. The component permits a specified kind of substance to move from one place to another.
2. **Influence**. The component tries to move a specified kind of substance. There are two subtypes according to the spatial relationship of the **influence** with potential sinks and sources.
   * **Pump**. The component tries to move a substance through it, e.g., a battery has a **pump** electricity behavior from the negative to the positive terminal. The sink and source are external to a **pump** behavior.
   * **Expel**. The component tries to move a substance from (or to) an internal container, e.g., a balloon has a **expel** air behavior.
3. **Move**. The component moves a specified kind of substance from one container to another along a specified path. **Move** behaviors are implicitly constrained by the amount and capacity of the containers.
4. **Create**. The component creates a specified kind of substance in a container, e.g., a light bulb has a **create** light behavior.
5. **Destroy**. The component destroys a specified kind of substance in a container, e.g., an acoustic insulator has a **destroy** sound behavior.

A behavior can be hypothesized based on causal patterns of behavior and structure. Its existence is confirmed, and its parameters are determined using knowledge about the physics of the substance being acted upon. Consolidation controls the inference of behavior by specifying the context (the composite component) in which inference can take place.

The causal patterns are similar to the process descriptions developed by Forbus [4]. Both identify the conditions necessary for some behavior to happen. One important difference is that the causal patterns are generic to all substances. While a process description can be stated at a high level of generality, there is no commitment by the theory to any particular level of generality. Another difference is that process descriptions state only how quantities change, while causal patterns apply to situations, such as two batteries connected in series, where no physical change takes place.

As an example, consider the device pictured in figure 2. The battery in the figure **pumps** electricity from one of its terminals to the other. It also **allows** electricity to flow between its terminals (otherwise the pumping action would have no effect). The switch **allows** electricity to flow through it when the state of the switch is closed. The light bulb also **allows** electricity to move through it, and **creates** light whenever electricity **moves** through it. The details of the representation and other behaviors of these components have been suppressed for explanatory purposes.

```
      +----------+
      |          |Pump electricity from - terminal
+----|-battery |     to + terminal
|     |    +     |Allow electricity between -
|     +----------+ terminal and + terminal
|           |
|           |
|           | end1
|      +---------+
|      |         |Allow electricity between end1
|      | switch  | and end2, state closed
|      |         |
|      +---------+
|           | end2
|           |
|           |end1  allow electricity between)
|    +-----------+ end1 and end2
|    |           |Create light in light bulb,
+---|light bulb|    dependency
end2|            |[move electricity between end1
     +-----------+       and end2
```

Figure 2. Light Bulb Device

During the consolidation process, a composite component consisting of the light bulb and the switch may be selected for processing. Because the switch's **allow** electricity behavior has a serial structural relationship with the light bulb's **allow** electricity behavior (satisfying the **serial allow** causal pattern), an **allow** electricity behavior from end1 of the switch to end2 of the light bulb is inferred. This **allow** behavior occurs only during the closed state of the switch, so the composite also has states of closed and open. General knowledge about electricity has an important role to play in this inference, specifically in determining the values of relevant attributes, such as electrical resistance, of the inferred **allow** electricity behavior. The switch-light bulb composite also has a **create** light behavior, which is "copied" from the light bulb's behavior description.

When this composite is combined with the battery, an **allow** electricity behavior around the circuit is inferred. Again, the **allow** behavior only occurs during the closed state. This behavior and the battery's **pump** electricity behavior satisfy another causal pattern, giving rise to a **move** electricity behavior around the circuit during the closed state. This **move** behavior satisfies the dependency of the **create** light behavior, thus the process infers that the device **creates** light while it is in the closed state.

In the inference of the creation of light, every behavior of the components and element of structure which plays some role in the creation of light has been used in the consolidation process. The explanation of this inference provides a complete causal account of the creation of light in the light bulb system in terms of the components' behavior and the device's structure.

We are implementing a version of consolidation, which will depend upon a few simplifying assumptions. The structural description will be limited to connection of components and containment of substances, thus reducing the amount of spatial reasoning required. Numerical attributes of behaviors (such as amount of influence or rate of movement) will be specified qualitatively. We hope to discover the limits of consolidation under these assump-

Appendix C

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

Technical Report

July, 1985

GENERIC TASKS IN EXPERT SYSTEM DESIGN AND THEIR ROLE IN EXPLANATION OF PROBLEM SOLVING

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

Invited presentation at the National Academy of Sciences / Office of Naval Research Workshop on AI and Distributed Problem Solving, May 16-17, 1985. To appear in the Proceedings of the Workshop.

Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving[1]

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

Abstract

We outline the elements of a framework for expert system design that we have been developing in our research group over the last several years. This framework is based on the claim that complex knowledge-based reasoning tasks can often be decomposed into a number of *generic tasks each with associated types of knowledge and family of control regimes.* At different stages in reasoning, the system will typically engage in one of the tasks, depending upon the knowledge available and the state of problem solving. The advantages of this point of view are manifold: (i) Since typically the generic tasks are at a much higher level of abstraction than those associated with first generation expert system languages, knowledge can be represented directly at the level appropriate to the information processing task. (ii) Since each of the generic tasks has an appropriate control regime, problem solving behavior may be more perspicuously encoded. (iii) Because of a richer generic vocabulary in terms of which knowledge and control are represented, explanation of problem solving behavior is also more perspicuous. We briefly describe six generic tasks that we have found very useful in our work on knowledge-based reasoning: classification, state abstraction, knowledge-directed retrieval, object synthesis by plan selection and refinement, hypothesis matching, and assembly of compound hypotheses for abduction.

[1]Research supported by Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010, and Air Force Office of Scientific Research grant 82-0255.

1. Information Processing Tasks in Knowledge-Based Reasoning

Intuitively one thinks that there are types of knowledge and control regimes that are common to diagnostic reasoning in different domains, and similarly there would be common structures and regimes for say design as an activity, but that the structures and control regimes for diagnostic reasoning and design problem solving will be generally speaking different. However, when one looks at the formalisms (or equivalently the languages) that are commonly used in expert system design, the knowledge representation and control regimes do not typically capture these distinctions. For example, in diagnostic reasoning, one might generically wish to speak in terms of malfunction hierarchies, rule-out strategies, setting up a differential, etc., while for design, the generic terms might be device/component hierarchies, design plans, ordering of subtasks, etc. Ideally one would like to represent diagnostic knowledge in a domain by using the vocabulary[2] that is appropriate for the task. But typically the languages in which the expert systems have been implemented have sought uniformity across tasks, and thus have had to lose perspicuity of representation at the task level. The computational universality of representation languages such as Emycin or OPS5 -- i.e., the fact that any computer program can be written in these languages, more or less naturally -- often confuses the issue, since after the system is finally built it is often unclear which portions of the system represent domain expertise and which are programming devices. In addition, the control regimes that these languages come with (in rule-based systems they are typically variants of *hypothesize and match*, such as forward or backward chaining) do not explicitly indicate the real control structure of the system at the task level. E.g., the fact that R1 [12] performs a linear sequence of subtasks -- a very special and atypically simple version of design problem solving -- is not explicitly encoded: the system designer so to speak "encrypted" this control in the pattern-matching control of OPS5.

These comments need not be restricted to the rule-based framework. One could represent knowledge as sentences in a logical calculus and use logical inference mechanisms to solve problems. Or one could represent it as a frame hierarchy with procedural attachments in the slots. (It is a relatively straightforward thing, e.g, to rewrite MYCIN [14] in this manner, see [16].) In the former, the control issues would deal with choice of predicates and clauses, and in the latter, they will be at the level of which links to pursue for inheritance, e.g. None of these have any natural connection with the control issues natural to the task.

[2]We also use the term *primitives of the language* in the rest of the paper to refer to the vocabulary.

Actually the situation is even worse: because of the relatively low level of abstraction relative to the information processing task, there are control issues that are artifacts of the representation, but often in our opinion misinterpreted as issues at the "knowledge-level." E.g., rule-based approaches often concern themselves with conflict resolution strategies. If the knowledge were viewed at the level of abstraction appropriate to the task, often there will be organizational elements which would only bring up a small, highly relevant pieces of knowledge or rules to be considered without any conflict resolution strategies needed. Of course, these organizational constructs could be "programmed" in the rule language, but because of the status assigned to the rules and and their control as knowledge-level phenomena (as opposed to the implementation level phenomena, which they often are), knowledge acquisition is often directed towards strategies for conflict resolution, whereas the really operational expert knowledge is at the organizational level.

This level problem with control structures is mirrored in the relative poverty of knowledge-level primitives for representation. E.g., the epistemology of rule systems is exhausted by data patterns (antecedents or subgoals) and partial decisions (consequents or goals), that of logic is similarly by predicates, functions, and related primitives. If one wishes to talk about types of goals or predicates in such a way that control behavior can be indexed over this typology, such a behavior can often be programmed in these systems, but there is no explicit encoding of them that is possible. E.g., Clancey [8] found in his work using Mycin to teach students that for explanation he needed to attach to each rule in the Mycin knowledge base encodings of types of goals so that explanation of its behavior can be couched in terms of this encoding, rather than only in terms of "Because <..> was a subgoal of <..>."

The above is not to argue that rule representations and backward or forward chaining controls are not "natural" for some situations. If all that a problem solver has in the form of knowledge in a domain is a large collection of unorganized associative patterns, then data-directed or goal-directed associations may be the best that the agent can do. But that is precisely the occasion for weak methods such as hypothesize and match (of which the above associations are variants), and, typically, successful solutions cannot be expected in complex problems without combinatorial searches. Typically, however, expertise consists of much more organized collections of knowledge, with control behavior indexed by the kinds of organizations and forms of knowledge in them.

To summarize the argument so far: There is a need for understanding the generic information processing tasks that underlie knowledge-based reasoning. Knowledge ought to be directly encoded at the appropriate level by using primitives that naturally describe the domain knowledge for a given generic task. Problem solving behavior for the task ought to be controlled by regimes that are appropriate for the task. If done correctly, this would simultaneously facilitate knowledge representation, problem solving, and explanation.

At this point it will be useful to make further distinctions. Typically many tasks that we intuitively think of as generic tasks are really *complex generic tasks*. I. e., they are further decomposable into components which ar more elementary in the sense that each of them has a homogeneous control regime and knowledge structure. For example, what one thinks of the diagnostic task, while it may be generic in the sense that the task may be quite similar across domains, it is not a unitary task structure. Diagnosis may involve classificatory reasoning at a certain point, reasoning from one datum to another datum at another point, and abductive assembly of multiple diagnostic hypotheses at another point. Classification has a different form of knowledge and control behavior from those for data-to-data reasoning, which in turn is dissimilar in these dimensions from assembling hypotheses.

Thesis: Given a complex real world knowledge-based reasoning task, and a set of generic tasks for each of which we have a representation language and a control regime to perform the task, if we can perform an epistemic analysis of the domain such that (i) the complex task can be decomposed in terms of the generic tasks, (ii) paths and conditions for information transfer from the agents that perform these generic tasks to the others which need the information can also be established, and (iii) knowledge of the domain is available to encode into the knowledge structures for the generic tasks; then that complex task can be "knowledge-engineered" successfully and perspicuously. Notice that an ability to *decompose* complex tasks in this way brings with it the ability to *characterize* them in a useful way. We can see, e.g., that the reason that we are not yet able to handle difficult design problem solving is that we are often unable to find an architecture of generic tasks in terms of which the complex task can be constructed.

In the rest of this paper, we will briefly describe some of the elementary generic tasks that we have had occasion to identify and use in the construction of expert systems. While we have been adding to our repertoire of elementary generic tasks over the years, the basic elements of the framework have been in place for a number of years. Our work on MDX [4, 5], e.g., identified *classification*, *knowledge-directed information passing*, and *hypothesis matching* as three generic tasks, and showed how certain classes of diagnostic problems can be implemented as an integration of these generic tasks. (We have earlier referred to them as *problem solving types*, but in [6], we began to call them generic tasks.) Over the years, we have identified several others: *object synthesis by plan selection and refinement* [1], *state abstraction* [7], and *abductive assembly of hypotheses* [11]. There is no claim that these are exhaustive; in fact, our ongoing research objective is to identify other useful generic tasks and understand their knowledge representation and control of problem solving.

2. Some Generic Tasks

2.1. Characterization of Generic Tasks

Each generic task is characterized by the following:

1. A task specification in the form of generic types of input and output information.

2. Specific forms in which the basic pieces of domain knowledge is needed for the task, and specific organizations of this knowledge particular to the task.

3. A family of control regimes that are appropriate for the task.

From the nature of the control regime, we can determine the *types of strategic goals* the problem solving for the task has. These goal types will play a role in providing explanations of its problem solving behavior.

When a complex task is decomposed into a set of generic tasks, it will in general be necessary to provide for communication between the different structures specializing in these different types of problem solving. Note that a decomposition does not imply that there is a predetermined temporal ordering on when the generic tasks are performed: typically the agent for a generic task is invoked when another agent needs information that the former can provide. Further there is no implication that there is a unique decomposition. Depending upon the availability of particular pieces of knowledge, different architectures of generic tasks will typically be possible for a given complex task.

We will now proceed to a brief characterization of these generic tasks.

- *I. Classification*

  Task specification: Classify a (possibly complex) description of a situation as an element, as specific as possible, in a *classification hierarchy*. E.g, classify a medical case description as an element of a disease hierarchy.

Forms of knowledge: <partial situation description> ---> evidence/belief about confirmation or disconfirmation of classificatory hypotheses. E.g., in medicine, a piece of classificatory knowledge may be: certain pattern in X-ray & bilirubin in blood ---> high evidence for cholestasis.

Organization of knowledge: The above classificatory knowledge *distributed* among concepts in a classificatory concept hierarchy. Each conceptual "specialist" ideally contains knowledge that helps it determine whether it (the concept it stands for) can be ***established*** or ***rejected***. The form of the knowledge as stated above is the form needed for this decision.

Control Regime: (Simplified form) Problem solving is top down. Each concept when called tries to establish itself. If it succeeds, it lists the reasons for its success, and calls its successors, which repeat the process. If a specialist fails in its attempt to establish itself, it rejects itself, and all its successors are also automatically rejected. This control strategy can be called *Establish-Refine*, and results in a specific classification of the case. (The account is a simplified one. The reader is referred to [5] for details and elaborations.)

Goal types: E.g., Establish <concept>, Refine (subclassify) <concept>

Example Use: Medical diagnosis can often be viewed as a classification problem. In planning, it is often useful to classify a situation as of a certain type, which then might suggest an appropriate plan.

- *II. State abstraction*

Task Specification: Given a change in some state of a system, provide an account of the changes that can be expected in the functions of the system. (Useful for reasoning about consequences of actions on complex systems.)

Form of knowledge: <change in state of subsystem> ---> <change in functionality of subsystem = change in state of the immediately larger system>

Organization of Knowledge: Knowledge of the above form distributed in conceptual specialists corresponding to system/subsystems. These conceptual specialists are connected in a way that mirrors the way the system/subsystem is put together.

Control regime: Basically bottom up, but follows the architecture of the system/subsystem relationship. The changes in states are followed through, interpreted as changes in functionalities of subsystems, until the changes in the functionalities at the level of abstraction desired are obtained.

Goal Types: E.g., Abstract consequent state, Deduce change in functionality.

Example Use: Answering questions of the form: "What will happen if this valve is closed, while the turbine is running?" Generic usefulness is in consequence finding.

- *III. Knowledge-Directed Information Passing*

Task specification: Given attributes of some datum, it is desired to obtain attributes of some other datum, conceptually related to the original datum.

Forms of Knowledge: i. Default value of <attribute> of <datum> is <value> ii. <attribute> of <datum> is inherited from <attribute> of parent of <datum> iii. <attribute> of <datum> is related as <relation> to <attribute> of children of <datum>. iv. <attribute> of <datum> is related as <relation> to <attribute> of <concept>.

Organization of Knowledge: The concepts are organized as a *frame hierarchy*. Default for slots corresponds to form i. above, the IS-A or PART-OF links between parents and children determine the types of inheritance in form ii. and iii. Procedural attachments or "demons" are used to encode form iv. Each frame is a specialist in knowledge-directed data inference for the concept.

Control regime: A concept, when asked for the value of one of its attributes first checks the data base to see if the actual value is known, then uses inheritance relationships to determine if the value can be obtained by inference from the values of appropriate attributes of its parent or children, then uses any demons that may be attached to the slot to query other concepts in other parts of the hierarchy for values of their attributes. If none of it succeeds and if it is appropriate the default value is produced as the value.

This is basically a hierarchical information-passing control regime, with demons providing an override of the hierarchical regime.

Goal Types: E.g., Inherit value of <attribute>. Ask for <concept, attribute value) to infer <attribute> by <relation>.

Example Use: Knowledge-based data retrieval tasks in wide variety of situations. Inferring a medical datum from another, when the latter is available but the former is needed for diagnostic reasoning. E.g., diagnostic reasoning needs information about whether the patient has been exposed to "anesthetics," because it has diagnostic knowledge that relates a diagnostic conclusion to this datum, but the patient data do not include any reference to "anesthetics," but mentions "major surgery a few weeks before." Assuming that the knowledge base for the data retrieval system encodes the piece of knowledge that relates "surgery" and "possible exposure to anesthetics," performing the reasoning that connects the two data items is an example of knowledge-based data retrieval.

- *IV. Object Synthesis by Plan Selection and Refinement*

Task Specification: Design an object satisfying specifications (object in an abstract sense: they can be plans, programs, etc.).

Forms of knowledge: Object structure is known at some level of abstraction, and pre-compiled plans are available which can make choices of components, and have lists of concepts to call upon for *refining* the design at that level of abstraction.

Organization of Knowledge: Concepts corresponding to "components" organized in a hierarchy mirroring the object structure. Each concept has plans which can be used to make commitments for some "dimensions" of the component,

Control Regime: Top down in general. The following is done recursively until a complete design is worked out: A specialist corresponding to a component of the object is called, the specialist chooses a plan based on some specification, instantiates and executes some part of the plan which suggests further specialists to call to set other details of the design. Plan failures are passed up until appropriate changes are made by higher level specialists, so that specialists who failed may succeed on a retry.

Goal Types: E.g., Choose plan, execute <plan element>, refine <plan>, redesign (modify) <partial design> to respond to failure of <subplan>S, select alternative plan, etc.

Example: Expert design tasks, synthesis of everyday plans of action.

- *V. Hypothesis Matching*

Task Specification: Given a hypothesis and a set of data that describe the problem state, decide if the hypothesis matches the situation.

Form and Organization of Knowledge: (One form) A hierarchical representation of evidence abstractions, top node is the degree of matching of the hypothesis to the data, and nodes at a given level are components of evidence for the evidence abstraction at the higher level. E. g., say the hypothesis of *goodness* of a position in a game is the one to be matched against the data describing the board configuration. Goodness may be defined at the top level in terms of two abstractions: *defensibility* and *offensive opportunities*. Form of knowledge then for this must be such as to enable mapping degrees of belief in each of these evidence abstractions to degree of belief in the *goodness* abstraction. The *defensibility* abstraction, e.g., may in turn be defined either by direct data or intermediate abstractions. Samuel's *signature tables* can be thought of as performing this task.

Goal types: Evaluate evidence for hypothesis, evaluate evidence for contributing abstraction .

- *VI. Abductive Assembly of Explanatory Hypotheses*

Task Specification: Given a situation (described by a set of data items) to be explained by the best explanatory account, and given a number of hypotheses, each associated with a degree of belief and each of which offers to explain a portion of the data (possibly overlapping with data to be accounted for by other hypotheses), construct the best composite hypothesis out of the given hypotheses.

Forms of Knowledge: causal or other relations (such as incompatibility, suggestiveness, special case of) between the hypotheses, relative significance of data items.

Organization of Knowledge: For relatively small number of hypotheses, this is a global process. For large numbers, some form of recursive assembly will be called for, implying knowledge organized at different levels of abstraction of the assembled hypotheses.

Control Regime: (Simplified version; see [11] for a fuller discussion.) Assembly and criticism alternate. In assembly, a means-ends regime, driven by the goal of explaining all the significant findings, is in control. At each stage, the most significant datum to be explained results in the best hypothesis that offers to explain it being added to the composite hypothesis so far assembled. After each assembly, the critic removes explanatorily superfluous parts. This loops until all the data are explained, or no hypotheses are left.

Goal Types: e.g, account-for <datum>, check-superfluousness-of <hypothesis>.

Example Use: In medical diagnosis, the classification generic task may produce a set of classifications, each of which accounts for some of the data. The best account needs to be put together. The Internist system [13] and the Dendral system [2] perform this type of task as part of their problem solving.

3. Encoding Knowledge at the Level of the Task

For each generic task, the form and organization of the knowledge directly suggest the appropriate representation in terms of which domain knowledge for that task can be encoded. Since there is a control regime associated with each task, the problem solver can be implicit in the representation language. I.e., as soon as knowledge is represented in the shell corresponding to a given generic task, a problem solver which uses the control regime on the knowledge representation created for domain can be created by the interpreter. This is similar to what representation systems such as EMYCIN do, but note that we are deliberately trading generality at a lower level to specificity, clarity, richness of ontology and control at a higher level.

We have designed and implemented representation languages for a simpler versions of two of these generic tasks: classification [3], and object synthesis by selection and refinement [1]. We plan to implement a family of such representation languages.

4. Generic Tasks and Explanation of Problem Solving

We have developed a framework for providing explanations for the decisions recommended by expert systems, and this is the basis of a four-year research effort sponsored by the Defense Advanced Research Projects Agency. For the purpose of this discussion, we can say that understanding the problem solving behavior of an expert problem solving system requires inspecting three structures, each corresponding to a type of explanation:

Type 1: trace of run-time, data-dependent problem solving behavior, viz.. which pieces of knowledge were used and how. E.g., Why do you say that the patient has cholestasis? Ans: Because the patient has high bilirubin in blood, and jaundice and Xrays suggest an obstruction in the biliary duct. This is typically done by checking which data items in the current case matched the piece of knowledge that enabled the system to make a particular decision.

Type 2: understanding how a piece of knowledge relates to the domain. how it can be justified. E.g., Why do you say that high bilirubin in blood suggests cholestasis? Here whatever answer is given, no data about the current case are being used. The system is really being called upon to justify its knowledge.

Type 3: understanding the control strategy used by the program in a particular situation. E.g. Why didn't you consider portal hypertension in this case? Ans: Because I had ruled out circulatory diseases, portal hypertension is a special case of circulatory diseases, and *my strategy is not to consider special cases when I have ruled out the general case.*

The explicit encoding of the generic aspects of knowledge and control behavior for each generic task can be directly used to produce explanations of Type 3. We will give some examples

Classification

Q: Why do you wish to know if the patient had been exposed to anesthetics?

A: Because I was trying to *establish* hepatitis.

Q : Why?

A: Because I had already established that the patient had liver disease. Now I am trying to subclassify the case, and hepatitis is one of the subcategories. If I can *reject* hepatitis, I can reject all the following diseases which are its subclasses.

State Abstraction

Let us assume that the user had asked the system, "What will happen if Valve A is closed?"

Q: Why do wish to know the engine speed?

A: Because I am trying to decide the *change in functionality of* the cooling system when Valve A is closed, and engine speed is a datum I need for this.

Q: Why are you considering the cooling syst^m?

A: Because, Valve A is a *component of* the Cooling system.

Data Retrieval

Q: Did the patient have exposure to anesthetics?

A: Did he have any surgery?

Q: Why do you ask?

A: I do not have direct record of anesthetic exposure. I failed to *inherit* a value for this attribute from its parent -- no direct record of any drug not being given, I also failed to infer No Exposure, by *ruling out its children.* Then the Anesthetic concept *suggested* that If surgery had been performed, anesthetic can be reasonably inferred.

Hypothesis Assembly

Q: Why was hypothesis part H' included in the best explanation?

A: In order to *account-for* <datum>

Q: Why wasn't H" chosen to explain D?

A: Because assuming <partially assembled conclusion>, H' is the best way to explain <cluster of data>.

Q: Why was hypothesis H accepted?

A: Because it is the only plausible way to *account-for* <cluster of data>.

Plan Refinement

Q: Why did you choose Plan A'?

A: Because, I am trying to complete the specification for Plan A, for *refining* which I need <subgoal> accomplished. The specialist for <subgoal> selected Plan A' due to <reasons>.

Q: What will you do if you fail in Plan A'?

A: <Subgoal> specialist will *select* Plan A''.

Q: What if it fails?

A: Parent specialist will *redesign* Plan A, by weakening <constraint>.

In the foregoing examples, the italicized terms represent the type of goal that is being pursued. Points to be noted here are: this explanatory richness (compared to the terminology of goal-subgoals) is made by possible by encoding the control regimes specific to each generic task; and, the explanation is directly related to the problem solving of the system.

4.1. Comparison with Related Work

With respect to providing explanation there are two key ideas that we are offering in this paper: one, explanation of problem solving strategies, which are manifested as appropriate control behavior by the problem solver, can be based on the generic task that a problem solver is engaging at a given stage in problem solving; and two, which is implicit in what we have said so far, is that control for each task be represented abstractly so that explanations can be couched in terms of these abstractions.

Swartout and Clancey have done significant investigations of issues in explanation generation by problem solving systems. The work of both authors uses the notion of *abstract representation of control* as a basic idea for explanation. It will be useful to relate our ideas to those of these investigators.

4.1.1. The Work of Clancey's Group:

Clancey has contributed several ideas that are relevant in this context: one, in [9], he discussed the advantages of abstract representation of control in reasoning systems, and specifically pointed out their potential role in explanation; two, in [8], he proposed that, in order to give explanatory capabilities to MYCIN for purposes of teaching (he created a system called GUIDON based on MYCIN) an explanatory skeleton be attached to each rule encoding the role of the rule in problem solving; and three, in his work on NEOMYCIN [10], he and his group represent the diagnostic strategy explicitly (in terms of abstract subtasks and their relations to diagnosis on the one hand and to the domain data on the other).

The most advanced work by Clancey's group on explanation is that on

NEOMYCIN, and thus we will concentrate on that in this section. Here diagnostic strategy is represented explicitly as a collection of subtasks, with conditions for moving from subtask to subtask also explicitly stated. This representation enables an explanation of strategy to be produced at the task and sub-task level of generalization.

This work is in many ways quite close in spirit to our approach, with the following comments throwing light on the differences.

1. NEOMYCIN's representation of abstract strategies is implemented as a body of metarules in the rule-based paradigm. We would note here that the rule paradigm plays no intrinsic role in this and can be viewed as *merely* an implementation language. In our approach we would advocate a representation language with generic primitive terms for directly encoding control along the lines discussed earlier in the paper.

2. The above comment raises the question of the appropriate language in which couch the tasks abstractly. In this paper we have proposed a set of generic tasks and suggested that they (and others to be added as needed on empirical grounds, but at about the same level of grain size) comprise the elementary tasks in terms of which complex (generic) tasks such as diagnosis be decomposed. While we have been able to demonstrate this claim to a certain extent for the diagnostic strategy employed by the MDX system, it is a matter of further empirical research to see whether and how NEOMYCIN's diagnostic strategy be so decomposed.

With respect to point 2 above, are there advantages from an explanation point of view for such a decomposition even if it were possible? At this point we can only give the following tentative answers. To the extent that the subtasks in NEOMYCIN were developed by a direct study of the diagnostic task, it is likely that some of these tasks (and consequently the terms which they contribute to the explanation) are more informative at the diagnostic task level. But if our theory is right, the additional abstractions specific to diagnosis can be obtained naturally from the abstraction at the generic task level. The generic tasks in our sense will have the further advantage of providing the primitives for other "molecular" tasks in addition to diagnosis.

4.1.2. Swartout and the XPLAIN System:

Swartout's XPLAIN system [15] can be summarized for our purposes as follows. It has a component called Domain Principles, which is best thought of as a base of *control abstractions* of the goal-subgoal type. They are of the form, "If goal is G, and if <pattern1>, ... <patternN> occur in the domain knowledge base, set up subgoals SG1, ... SGN respectively." As a concrete example, G might be "Administer <drug>," pattern1 might be, "<finding> and <drug> cause <bad side effect>," and SG1 might be, "Control toxicity of <drug>." One can imagine an instructor teaching a group of students about administration of drugs in general, and telling them that if, for a particular drug, there is a possibility of a bad side effect, then make sure to do whatever will be needed to control the drug toxicity. Note that this has some degree of generality in that it can be used to set up systems for a number of different drugs: if a certain drug does not cause bad side effects, then this particular subgoal will not be set up by the system. In general one can best think of this approach as specification of an *expert system generator*, in that the same Domain Principles base can be used to generate, e.g., systems to recommend the administration of different drugs. The Domain Principles then can be thought of as a collection of control abstractions. However, these control abstractions are domain-specific. Terms such as *administer* and *control toxicity* in the example above are used to index and name goals, but do not have general purpose problem solving relevance across domains. The only elements in the above example that *are* generic in our sense are, *If goal*, and *set up subgoal...*

As one would expect, the basis for the explanation capability of XPLAIN arises from the goal-subgoal control abstractions in Domain Principles. The generation of explanation in XPLAIN is very similar to that in rule-based systems in that the goal-subgoal structure in Domain Principles is used for the explanation in a way very similar to the rule-tracing in backward-chaining systems such as Mycin. While explanation in Mycin is done using the trace of the rules that fired in a particular problem, XPLAIN uses the goal-subgoal relationships that went into the construction of the expert system, with very similar effects. XPLAIN can use the names of the goals and subgoals and the terms in the patterns to provide a richer quality to the explanation: "Because goal is to *administer* digitalis, and *digitalis causes dangerous side effects*, there is a need to *control toxicity of digitalis*."

Where our work differs from this effort is in the power that is available in the control abstractions that are indexed by generic tasks. This enlarges the kinds of explanations that can be provided in a domain-independent way, and that can arise directly from the control behavior in the problem solving process.

Acknowledgment: The paper has benefited from the comments of Tom Bylander, Jon Sticklen and John Josephson.

References

[1] Brown, D.C. / Chandrasekaran, B.
Expert Systems for a Class of Mechanical Design Activity.
1984
Paper for IFIP WG5.2 Working Conference. Sept. 84.

[2] Buchanan, B. / Sutherland, G. / Feigenbaum, E.A.
Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organcic Chemistry.
1969
in Machine Intelligence 4, American Elsevier, New York.

[3] Bylander, T. / Mittal, S. / Chandrasekaran, B.
CSRL: A Language for Expert Systems for Diagnosis.
In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 218-221. , August, 1983.
To appear in the Special Issue of Intn'l Jrnl. of Computers and Mathematics on "practical artificial intelligence systems".

[4] Chandrasekaran, B. / Mittal, S. / Gomez, F. / Smith M.D., J.
An Approach to Medical Diagnosis Based on Conceptual Structures.
Proceedings of the 6th International Joint Conference on Artificial Intelligence :134-142, August, 1979.
IJCAI79.

[5] Chandrasekaran, B. / Mittal, S.
Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems.
In M. Yovits (editor), *Advances in Computers*, pages 217-293. Academic Press, 1983.

[6] Chandrasekaran, B.
Expert Systems: Matching Techniques to Tasks.
1983
Paper presented at NYU symposium on Applications of AI in Business. Appears in Artificial Intelligence Applications for Business, edited by W. Reitman, Ablex Corp., publishers.

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

[8] Clancey, William J.
The Epistemology of a Rule-Based Expert System--a Framework for Explanation.
Artificial Intelligence 20(3):215-251, May, 1983.

[9] Clancey, William J.
The Advantages of Abstract Control Knowledge in Expert System Design.
In *Proceedings of AAAI-83*, pages 74-78. Amerian Association for Artificial Intelligence, 1983.

[10] Hasling, Diane Warner/ Clancey, William J./ Rennels.Glenn.
Strategic Explanations for a Diagnostic Consultation System.
Developments in Expert Systems.
London and New York: Academic Press, 1984, pages 117-133.

[11] Josephson, John R. / Chandrasekaran, B. / Smith, J.W.
Assembling the Best Explanation.
In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems.* IEEE Computer Society, Denver, Colorado, December 3-4, 1984.
A revised version by the same title is now available.

[12] McDermott, J.
R1: A Rule-Based Configurer of Computer Systems.
Artificial Intelligence 19, 1:39-88, 1982.

[13] Pople, H. W.
Heuristic Methods for Imposing Structure on Ill-Structured Problems.
Artificial Intelligence in Medicine.
Westview Press, .

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

[15] Swartout, W. R.
XPLAIN: A System for Creating and Explaining Expert Consulting Programs.
Artificial Intelligence 21(3):285-325, September, 1983.

[16] Szolovits, P./ Pauker, S. G.
Categorical and Probabilistic Reasoning in Medical Diagnosis.
Artificial Intelligence :115-144, 1978.

Appendix D

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

Technical Report

July, 1985

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

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

Invited paper presented at the Workshop on Pattern Recognition in Practice-II. Free University, Amsterdam, June 19-21, 1985. This paper will appear in the book, *Pattern Recognition in Practice-II*, E. S. Gelsema and L. N. Kanal, editors, North-Holland Publishing Co.

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

B. Chandrasekaran

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

In this paper we consider a very general information processing task: *classification*, and review the perspectives of the classical pattern recognition approaches and the more recent artificial intelligence/ knowledge-based systems point of view. As the complexity of the problem increases, we trace the evolution of the algorithms from numerical parameter setting schemes through those using symbolic abstractions and then relations between symbolic entities, and finally to complex symbolic descriptions which incorporate explicit domain knowledge.

1. INTRODUCTION

In this paper we consider a very general information processing task: *classification*, and review the perspectives of the classical pattern recognition approaches and the more recent artificial intelligence/ knowledge-based systems point of view. As the complexity of the problem increases, we trace the evolution of the algorithms from numerical parameter setting schemes through those using symbolic abstractions and then relations between symbolic entities, and finally to complex symbolic descriptions which incorporate explicit domain knowledge. The paper can be viewed two perspectives: as a bridge-building activity, describing the approaches of two 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 needed to evolve into complex symbolic processes in order to handle increasing complexity of problems and requirements for flexibility.

2. THE CLASSIFICATION TASK

2.1. Expert Systems and the Classification Task

The area of expert systems, though of recent origin, is already a well-established subarea of Artificial Intelligence. The essential idea of the field is an attempt to capture in computer programs, explicitly and in symbolic form, the *knowledge and problem solving methods* of human experts in selected domains and tasks; in fact, because of the central role of explicit domain knowledge, the field itself is often called *knowledge-based systems*. This is not an appropriate place to discuss the knowledge representation and problem solving issues in the field of expert systems, many of which are thriving and open research issues. There are many expert tasks that have been successfully emulated by these systems, while there are an even larger number of things that human experts do that are beyond the current state of the expert system technology. If one were to examine the intrinsic nature of the tasks of the current generation of expert systems, a surprising fact emerges: most of them solve variants of problems which are intrinsically *classificatory* in nature. It is important to note that we are not claiming that the authors of these programs *recognized* them as classification problems and used methods appropriate to that task, but that, independent of *how they were solved*, the problems solved by them have an intrinsically classificatory character.

Let us take some examples:

1. MYCIN [23], in its diagnostic phase, has the task of classifying patient data in an *infectious agent hierarchy*. I.e., the diagnostic task is identification of the infectious agent class as specifically as possible.

2. PROSPECTOR [10] classifies a geological description as corresponding to one or more mineral formation classes.

3. MDX [4, 5] explicitly views a significant portion of the diagnostic task as classifying a complex description (the patient data) as an element in a disease classification hierarchy (e.g., liver disease, in particular hepatitis).

4. SACON [2] classifies structural analysis problems into classes for each of which a particular family of analysis methods will be appropriate.

The above is by no means to imply that all problems are classification problems or that can be usefully converted into such problems. R1 [17], e.g., performs a relatively simple version of an *object synthesis* problem, i.e., a version of the design problem. RED [13] Internist [21] and Dendral [3] are different systems all performing various versions of *assembly of composite hypothesis* for *abductive reasoning*. In [6, 7, 8] we have given taxonomies of such *generic tasks* for which expert systems can be designed and we identified classification as one of the generic tasks. Recently Clancey [9] has made a similar assessment of how several expert systems perform classificatory problem solving. Sticklen, et al [25] discuss the control issues inherent in the task.

What is important to note from the above list is that classification seems to be a rather ubiquitous problem solving process, and a number of real world problems including medical diagnosis can be thought of as having a large classification component. Further, classification has been one of the more *tractable* problems for the knowledge-based system technology to handle at this point in its development.

2.2. Classification in Pattern Recognition

There is another area of inquiry, which is now more than 20 years old, viz., *pattern recognition*, which has also been intimately connected with problems of classification. In fact, in the early days of the field the problem of recognition was *formulated* as a problem of classification, in particular one of *statistical classification* of multidimensional *pattern vectors* into one of a finite number of classes, each class characterized by some kind of probability distribution. In fact what started out as a useful formulation practically got to be so dominant that there was a need for a paper such as that by Kanal and Chandrasekaran [14] 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*.

2.3. Classification in Biology

Taxonomic classification has long been a significant methodology in biology. Linnaeus's classification scheme is very famous, and more recently, mathematical taxonomy has been pressed into service for providing better classification in this area [24] The more recent controversies regarding evolutionary biology (the claddists vs traditional evolutionary theorists) revolve around implications of various theories for classification.

2.4. Why this Ubiquity? The Computational Power of Classification

Classification seems to be a powerful human method of organization for comprehension and action. This tendency is so strong that people often feel they have accomplished something by merely naming something as a class, even if they cannot do much about it. Why is classification so powerful?

A simple computational explanation can be given for the importance of classification as an *information processing strategy*. One can think of the task of an intelligent agent as performing actions on the world for certain goals. But often the correct action knowledge is a function of the state of the world. E.g., one can think of the general problem facing the physician as having the following formal character: For each subset of possible symptoms (the *state of the patient*), find an appropriate therapeutic action. But in general the cardinality of the relevant states of the world is too large: e.g., the total number of states of a patient is the cartesian product of the distinct states of each of the state variables (symptoms, laboratory values, manifestations of all kinds). A table relating the subset of state variables to action is bound to be too large for construction, looking up, and modification. This problem is made more tractable, however, if action knowledge can be *indexed*, not by the states of the world, but *by equivalence classes of states of the world*. Thus a physician's therapeutic knowledge is not indexed 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. The medical problem solving can then be organized first as mapping from symptoms to disease classes (diagnosis as classification), and then from disease classes to therapeutic actions. Since the number of equivalence classes is much smaller than the number of states, the complexity of the mapping is now considerably reduced.

Thus: classification into categories provides a great computational advantage. Much of human thinking is organized around classification, both in terms of creating useful classifications (concept learning) and using existing categories to perform classifications.

However, the process of creating useful classifications (concept learning) is a much harder process than using a classification structure to do the actual classification. Thus in medicine discovery of a disease (creation of a new class) is a relatively major event while diagnosis is much more routine. In this paper we only deal with the process of assigning an object to a class in a classification structure.

3. THE STATISTICAL CLASSIFICATION PARADIGM

The typical model in this paradigm is one where the aim is to arrive at a classification of a multidimensional vector (where each dimension is typically a numerical variable, even though ordinals are some times used) representing an object of unknown classification into one of a finite number of classes. Each dimension typically represents an *attribute* of the object that the system designer has had reason to believe carries useful information about class membership. Intuitively, one would try to choose attributes such that they have the potential to *distinguish* between classes. When the number of dimensions is small, it is possible to design classification systems that outperform human expert performance in that task domain. E.g., if it is desired to distinguish between diseases D1 and D2, and statistical evidence indicates that symptoms s1, s2, ..sN carry useful information for this discrimination, careful statistical data gathering is often possible such that a discriminant function of the variables s1... sN can often be a very accurate classifier. Human reasoning with the same variables may be less efficient in information extraction, and thus automatic procedures using the statistical model can be very powerful.

In this model, in spite of the enormous intrinsic interest of the mathematical problem of designing and improving classification algorithms -- virtually thousands of papers have been written in the pattern classification literature on solutions to this problem -- Kanal and Chandrasekaran [14] pointed out years ago that the real power often comes from the careful choice of the variables themselves based on a good knowledge of the domain rather than from the complexity of the separation algorithm.

What happens when the dimensionality of the vector gets to be very large, or the number of classes gets to be large? When the number of classes increases, in general, in order to make more and more distinctions the number of measurements on the object, i.e., the dimensionality of the pattern vector, also will need to grow rapidly. When the numbers of classes increases, the complexity of the algorithm to make the discrimination grows much more rapidly, and correspondingly the average performance, i.e., correct classification rate, deteriorates quite rapidly. Sensitivity problems begin to become quite severe: i.e., the required precision of the parameters in the classification algorithm becomes impractically high. Opacity problems result: it becomes increasingly hard to make any kind of statement about what attributes are playing what role in the recognition process. These problems exist whether statistical classification algorithms are used, or perceptron-like linear threshold devices are used. Szolovits and Pauker [26] discuss some of the problems with the Bayesian approaches, and Minsky and Papert [18] the problems with the latter.

4. ABSTRACTION BY INTERMEDIATE CONCEPTS

What is to be done when the number of classes is very large? Consider the following pedagogically useful example: the design of recognition devices for automatic reading of texts. Assume for the sake of discussion that the number of words in the language is 20,000, and we would like the words to be recognized. Consider solving the problem by designing a discriminant function which directly maps a multidimensional vector into one of 20,000 classes. One can sense that this is a pretty unworkable solution: the number of measurements that would need to be made on the words and the complexity of the decision algorithm will be too large to permit this solution in practice. Intuitively one would think that recognizing characters first, and then based on this recognition recognizing words would be computationally more attractive. Why is this a much more reasonable solution?

What is going on here is a two-fold strategy of *symbolization* and *hierarchicalization*. Instead of doing the classification by a direct discriminant function-like mapping, intermediate symbols are constructed, which are then used as attributes to a higher-level classification process. (Note that this is not the same as *hierarchical classification*, which if it were to be applied to this problem, will first involve classification of the words into groups of similar-looking words, each class will be further subclassified, and so on, until each word receives a classificatory status.) Symbols at each level are produced by a classificatory process using the symbols from the previous level as attributes. Each such computational process is much more tractable.

4.1. Signature Tables

Consider another example: evaluation functions in chess. 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 for the evaluation functions was a linear polynomial of attributes of the board: both the attributes and their weights were chosen in consultation with domain experts. Then in order to take into account interactions between the variables in the evaluation function, higher order polynomials were later proposed. This of course resulted in a fairly rapid increase in the complexity of the function: if r-the order interactions were to be included and the number of attributes is n, then the number of terms was of the order of n**r. Samuel's *signature tables* [22] provided a solution which exemplifies the symbolization and hierarchicalization ideas mentioned earlier. 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 final abstraction, in this case, a measure of the "goodness" of the board. (Typically the attributes in a group may have some dependencies and interactions, in order to capture which, in the more traditional evaluation functions, polynomial term were included.) In chess, "defensibility of king" and "material advantage" may be such intermediate concepts, each of which can be estimated by a subset of board attributes, while the final decision about the goodness of a board configuration may be made in terms 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. (For the purpose of our discussion, the exact method is not important-- Samuel proposed a specific mechanism for this. The essence of his mechanism is a mapping from a multidimensional vector, each component of which can only be in one of a small number of distinct values, to a symbolic abstraction, which can also be in only one of a small number of distinct values. The mapping turns out to be a fairly simple one.)

3. The outputs of the classifiers for each group can themselves be thought of as qualitative attributes at the next level of abstraction. These can be grouped and abstracted into higher level concepts as necessary until the top-level concept is a classification of the "goodness" of the board in a qualitative way.

To repeat a point made earlier, by trading off the precision of numbers for the simplicity and robustness of a small number of symbolic states, and combining it with hierarchical abstractions, significant computational advantage is being gained. It also points to the fact that often numbers are too precise for the task at hand. Robust symbolic abstractions of the appropriate kind can capture almost all of the relevant information.

5. SYNTACTIC OR STRUCTURAL APPROACHES

After about a decade of work within the statistical classification paradigm (the work on the perceptron paradigm was going on in parallel in certain sections of the AI community), Narasimhan [20] proposed what he called a *syntactic approach* to pattern recognition. The idea was to describe classes of patterns, not in terms of probability distributions in multidimensional spaces, nor in terms of hierarchic symbolic abstractions, but in terms of *relations between symbols*, much as grammatical categories are described in linguistic analysis. Most of the readership of this book will be familiar with the principles behind and examples of syntactic methods for pattern recognition, so we forego a description here. (By now there is a vast literature on the subject.) The following point, however, can be noted: *The ability to describe a class in terms of relations is a move towards descriptions as the basis for class characterization.*

Note that the idea of syntactic pattern recognition is really a special case of the more general notion of *structural relations* as the basis of class characterization. (In [15] we discuss the relationship of the structural paradigm to the statistical one.) Thus, even when the idea of *syntax* is not appropriate -- it is very doubtful that the notion of a picture grammar is as general for the domain of visual objects as seems from a purely formal perspective -- the notion of structural relations as the basis for characterizing concepts and classes is a somewhat more general one.

With the introduction of *syntactic/structural models* for pattern recognition), the progression becomes:

numbers --> symbols --> relations.

The major research directions in pattern recognition for capturing structural relations in general were *formal*, i.e., some sort of a mathematical system within which theorems about relationships 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. In any case, the emphasis on formalisms led to two constraints: one, often an attempt was made to force-fit available formalisms to the pattern recognition problem, generally with unsatisfactory results; and two, because human classification performance was more heuristic in nature, restricted formalisms could only capture the quality of human performance only fleetingly.

If one is to use relations between symbolic attributes as the basis of class characterization, why restrict oneself to *syntactic* relations? Why not bring to bear the full power, to the extent possible or necessary, the *semantics* of the classes in forming class descriptions? Asking this question prepares the way for the next step in the progression, the AI/Knowledge-based paradigm:

numbers --> symbols --> relations --> complex symbolic descriptions.

These complex descriptions characterizing the classes are the relevant aspects of the domain knowledge for the task.

6. AI/KNOWLEDGE-BASED REASONING: A NEW PARADIGM

It is not an exaggeration to say that the knowledge-based approach in general, and to classification in particular, is a *new paradigm* in the sense that it emphasizes different issues, and poses them in a different language. E.g., instead of issues such as "optimality" and "error rate," which figure in the classical pattern recognition approach, the AI paradigm emphasizes the issues of "knowledge representation" and "control of problem solving". These relate to how the domain knowledge in explicit symbolic form is *represented* (i.e, what language is used to encode the knowledge), *organized* and *accessed*, and how the knowledge is *used* to arrive at classificatory conclusions.

We have already used medical diagnosis as an example to illustrate some of the ideas in this paper. In discussing the AI approach, the medical diagnosis example is particularly useful, since a number of AI systems have been built in this domain, and also the computational advantages of such an approach can be well motivated. We will briefly describe a system called MDX [5] [12] which has been developed in our Laboratory over the past several years. Our description will necessarily be brief, since our aim is to point to the role of knowledge structures and to give a feeling for what characterizes the AI approach.

6.1. The MDX System: Example of Knowledge-Based Approach to Classification

The MDX system performs medical diagnosis by essentially viewing the task as one of classifying a complex case description as a node in a disease classification hierarchy. A number of caveats need to be kept in mind:

1. Not all classification problems are necessarily solved as *hierarchical classification problems*. There are other AI systems that perform classification, but without using the hierarchical point of view: e.g., [1].

2. In general multiple classification hierarchies may exist in domain. (E.g., in medicine, "viral hepatitis" is a classificatory concept in the "infectious disease" hierarchy as well as in the "liver disease" portion of the hierarchy.) The general problem involves coordinating among the classifications by the different classification systems.

3. A particular case may not have a single classification, but instead have several classifications simultaneously applicable. (E.g., a patient may have both "cirrhosis" and "portal hypertension," and in addition, the two diseases may be causally related. This sort of situation may also arise in other domains: in character recognition, the image may really be two characters touching each other, e.g., rather than one character.) The MDX framework can deal with many of these complexities more or less well, but for the purpose of this paper we will concentrate on the single classification situation.

The *control problem* here can be stated as one that deals with what classificatory hypothesis to consider at what point in problem solving. In general we would like to use domain knowledge to consider only a subset of all the hypotheses for problem solving efficiency, or we would like to consider some hypotheses which are more promising ahead of others.

The MDX system is organized as a hierarchical collection of or "diagnostic concepts," each of which has diagnostic knowledge that helps it make a determination about the relevance of that hypothesis (at that level of abstraction) to the case at hand. This hierarchy of specialists mirrors the diagnostic classification hierarchy. The total diagnostic knowledge is then distributed through the conceptual nodes of the hierarchy in a specific manner to be discussed shortly. The problem-solving for this task will be performed top down, i.e., the top-most concept will first get control of the case, then control will pass to an appropriate successor concept, and so on. In the medical example,

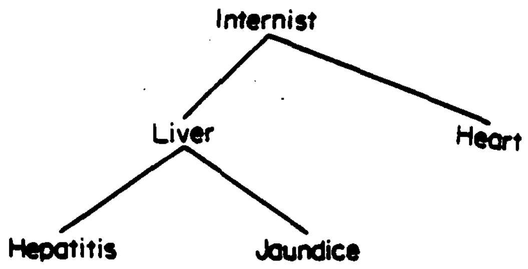


Figure 1: **Fragment of a hierarchy**

a fragment of such a hierarchy might be as shown in Fig. 1. More general classificatory concepts are higher in the structure, while more particular ones are lower in the hierarchy. It is as if INTERNIST first establishes that there is in fact a disease, then LIVER establishes that the case at hand is a liver disease, while say HEART etc. reject the case as being not in their domain. After this level, JAUNDICE may establish itself and so on.

Each of the concepts in the classification hierarchy has "how-to" knowledge in it in the form of a collection of *diagnostic rules* (This is only one possible method by which the specialists can make the determination about their fit with the data. In simple cases, statistical classification algorithms can be used. In DART [11] the decision about the fit of hypothesis to data is done by using theorem-proving techniques. In [19], we show how the concepts can make their decisions based on a causal knowledge of the domain. 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 the hypothesis matches the case at hand). These rules are of the form: <symptoms> ---> <concept in hierarchy>, e.g., "If high SGOT, add n units of evidence in favor of cholestasis." Because of the fact that when a concept rules itself out from relevance to a case, all its successors also get ruled out, large portions of the diagnostic knowledge structure never get exercised. On the other hand, when a concept is properly invoked, a small, highly relevant body of knowledge comes into play.

The problem-solving that goes on in such a structure is *distributed*. The problem-solving regime that is implicit in the structure can be characterized as an *establish-refine* type. That is, each concept first tries to establish or reject itself. If it succeeds in establishing itself, the refinement process consists of seeing which of *its* successors can establish itself. Each concept has several clusters of rules: confirmatory rules, exclusionary rules, and perhaps some recommendation rules. The evidence for confirmation and exclusion can be suitably weighted and combined to arrive at a conclusion to establish, reject or suspend it. The last mentioned situation may arise if there is not sufficient data to make a decision. Recommendation rules are further optimization devices to reduce the work of the subconcepts. Further discussion of this type of rules is not necessary for our current purpose.

The concepts in the hierarchy are clearly not a static collection of knowledge. They are active in problem-solving. They also have knowledge only about establishing or rejecting the relevance of that conceptual entity. Thus they may be termed "specialists," in particular, "diagnostic specialists." The entire collection of specialists engages in distributed problem-solving.

The MDX system 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 specialists. The number of symptoms, signs, and laboratory values that it can handle is in the hundreds, and the number of distinct hypotheses it has in its diagnostic hierarchy is also close to hundred. Some of the laboratory results about images is in complex descriptive form. Hard probability numbers are nowhere used: what the specialists compute can be thought of as qualitative probability values: "definitely present," "likely present,"... "definitely absent." This is the sort of problem for which a purely numerical mapping approach such as a Bayesian one will have considerable computational problems, in addition to posing difficulties of knowledge acquisition. It is often quite difficult to acquire probability distributions of the type needed for the classification algorithms from physicians, at least *for problems of this degree of complexity, while the sort of knowledge MDX uses is* directly available from domain experts.

7. CONCLUDING REMARKS

It is by now a truism that significant aspects of thinking can be modeled as symbolic information processing: creation and manipulation of complex symbolic structures bearing knowledge of various of types. Artificial intelligence deals with such models couched in explicitly computational terms. We have noted that classification seems to be an ubiquitous method used by human thought processes, and pointed out *that the reason for that* is the significant computational advantages that arise from storing knowledge useful for action by indexing it 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 for classification: numerical measures and formulae of various types, symbolic abstractions, hierarchical symbol structures, structural relations between symbols, and finally to rich symbolic knowledge structures. Each stage in the progression gave more power in controlling the computational complexity by matching the structure of the classifier to the complex structure of the task. At the level of knowledge, the power comes from *task-specific control regimes* controlling access to *appropriate chunks of knowledge*. We motivated the discussion by using medical diagnosis as an example in various places, but the ideas are more generally applicable.

The discussion in this paper can be viewed as a bridge-building activity between two research paradigms: pattern recognition and artificial intelligence. Classification has been a major concern in the former, and an important task performed by many systems in the latter and thus the task provides a good place to understand the distinctions between the two research paradigms. *For well-constrained classification problems with* relatively small number of categories, the statistical and other numerical algorithms considered in the field of pattern recognition can provide powerful classifiers which often outperform human experts by extracting the last trace of information that the more discrete human symbolic processes can only approximate. On the other hand for complex problems involving many variables and classes the knowledge-based approach trades off the optimality of the best algorithms in pattern recognition for greater computational tractability and better matching with human knowledge in the domain.

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 point to the emergence of the need and power of symbolic structures for control and prediction. Cybernetics showed the power and usefulness of the concepts of feedback and stability in understanding many control and communication problems. But in classical control theory, numbers and functions hold sway. 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 limited. Symbolic models of the world provide greater leverage for change and control and still keep computational costs within manageable bounds. Thus in biological information processing, symbolization seems to have occurred very early in evolution: see [16] for an account of how early visual processing of the frog is symbolic in nature. 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 points in this paper can be viewed as an intuitive account of the emergence and power of symbolic structures for complex information processing activities.

Our approach within artificial intelligence has been to identify other *generic tasks* similar to classification, but with similar characteristic of being a building block for complex information processing activities. In [8] we give an account of the latest repertoire of such generic tasks we have identified.

Acknowledgment

Research supported by grants AFOSR 82-0255 and NSF MCS-8305032 for developing these ideas is gratefully acknowledged.

References

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

[2] Bennett, J., Creary, L., Englemore, R., and Melosh, R.
SACON: A Knowledge-Based Consultant for Structural Analysis.
Sept 1978
STAN-CS-78-699 and HPP Memo 78-23, Stanford University.

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

[4] Chandrasekaran, B., Mittal, S., Gomez, F., and Smith M.D., J.
An Approach to Medical Diagnosis Based on Conceptual Structures.
Proceedings of the 6th International Joint Conference on Artificial Intelligence :134-142, August, 1979.
IJCAI79.

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

[6] Chandrasekaran, B.
Towards a Taxonomy of Problem-Solving Types.
AI Magazine 4(1):9-17, Winter/Spring, 1983.

[7] Chandrasekaran, B.
Expert Systems: Matching Techniques to Tasks.
1983
Paper presented at NYU symposium on Applications of AI in Business. Appears in *Artificial Intelligence Applications for Business*, edited by W. Reitman, Ablex Corp., publishers.

[8] Chandrasekaran, B.
Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving.
May 1985
Invited presentation at the National Academy of Sciences Office of Naval Research Workshop on AI and Distributed Problem Solving, May 16-17, 1985. To appear in the Proceedings of the Workshop. .

[9] Clancey, William J.
Classification Problem Solving.
In *Proceedings of the National Conference on Artificial Intelligence*, pages 49-55. AAAI, University of Texas at Austin, August 6-10, 1984.

[10] Duda, R.O., Hart, P.E., Nilsson, N.J., Reboh, R., Slocum, J., and Sutherland, G.L.
Development of a Computer-Based Consultant for Mineral Exploration.
October 1977
Annual Report, SRI Projects 5821 and 6415, SRI International, Menlo Park, California.

[11] Genesereth, M.R.
Diagnosis Using Hierarchial Design Models.
Proc. National Conf. on AI, American Association for Artificial Intelligence . 1982.

[12] Gomez, F.
Understanding Programming Problems Stated in Natural Language.
OSU CISR Tech Report.
February, 1981

[13] Josephson, John R., Chandrasekaran, B. and Smith, J.W.
Assembling the Best Explanation.
In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems.* IEEE Computer Society, Denver, Colorado, December 3-4, 1984.

[14] Kanal, Laveen and Chandrasekaran, B.
Recognition, Machine 'Recognition' and Statistical Approaches.
Methodologies of Pattern Recognition.
Academic Press, 1969, pages 317-332.

[15] Kanal, Laveen and Chandrasekaran, B.
On Linguistic, Statistical and Mixed Models for Pattern Recognition.
Frontiers of Pattern Recognition.
Academic Press, 1972, pages 163-192.

[16] Lettvin, Jerome, Maturana, H., McCulloch, W.S., and Pitts, W.
What the Frog's Eye Tells the Frog's Brain.
In *Proceedings of the IRE*, pages 1940-1951. 1959.

[17] McDermott, J.
R1: A Rule-Based Configurer of Computer Systems.
Artificial Intelligence 19, 1:39-88, 1982.

[18] Minsky, Marvin, and Papert, Seymour.
Perceptrons.
The MIT Press, 1969.

[19] Sembugamoorthy, V. and Chandrasekaran, B.
Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems.
August, 1984
To appear in *Cognitive Science.*

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

[21] Pople, H. W.
Heuristic Methods for Imposing Structure on Ill-Structured Problem.
Artificial Intelligence in Medicine.
Westview Press, 1982.

[22] Samuel, Arthur L.
Some Studies in Machine Learning Using the Game of Checkers II. Recent Progress.
IBM Journal of Research and Development 11(6), 1967.

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

[24] Sokal, R.R. and Sneath, P.H.A.
Principles of Numerical Taxonomy.
Freeman, W.M., San Francisco, 1963.

[25] Sticklen, Jon, Chandrasekaran, B. and Josephson, John R.
Control Issues in Classificatory Diagnosis.
In *Ninth International Joint Conference on Artificial Intelligence.* 1985.

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

Appendix E

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

Technical Report

July, 1985

A CRITIQUE OF QUALITATIVE SIMULATION FROM A CONSOLIDATION VIEWPOINT

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

Invited presentation at The IEEE International Conference on Systems, Man and Cybernetics, Tucson, Arizona, November 12-15, 1985. To appear in the Proceedings of the Conference.

A Critique of Qualitative Simulation from a Consolidation Viewpoint

Tom Bylander

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

Abstract

To understand commonsense reasoning, we need to discover what kinds of problems a commonsense reasoner should be able to solve, what the reasoner needs to have in order to solve those problems, and the relationships among the various kinds of problem solving abilities. We examine three methods for performing qualitative reasoning about the behavior of physical situations. Two of the methods perform qualitative simulation, which determines the behavior of a situation by a qualitative version of simulation methods. The other method is called consolidation, which derives the behavior of a situation by composing the behavior of the situation's components. We show that qualitative simulation and consolidation work on different problems of qualitative reasoning, and that their differences and similarities lead to several implications about their role in qualitative reasoning.

1. Introduction

A recurring criticism of knowledge-based systems, expert systems in particular, is that their knowledge is too "shallow." The criticism is directed at several symptoms which arise when the decisions made by these systems are based on rules of association instead of being based on a model of the domain. For example in MYCIN, "head injury" is used as evidence for "E. Coli causing meningitis," but MYCIN has no model of infectious diseases which supports this association, vis., a head injury exposes the meninges to bacteria in the environment, E. Coli is a common bacterium in the environment, and E. Coli in the meninges will often cause meningitis. Without this model, MYCIN is unable to explain the causal relationships between data and hypotheses; is unable to block associations when they are not appropriate, e.g., if the head injury occurred in a sterile environment; and is unable to give weight to hypotheses in slightly different, but similar circumstances, e.g., if other pathways to the meninges are available to E. Coli. Examples like these are not just endemic to MYCIN, but occur whenever associational knowledge is not supported by domain models.

The reason why association-based systems continue to abound is because robust models for domains as complicated as MYCIN's do not yet exist. One area of AI research which is directed towards this goal is *qualitative reasoning*, the ability to make decisions and solve problems based on qualitative data and models. The goal of qualitative reasoning is to achieve predictive and explanatory power similar to that of quantitative and analytical models while avoiding the need for precise formulations of problems and computationally-intensive methods. So in the situation where a flame is under a pan of water, one can predict that the water will probably heat up and boil. Even though only rough details of this situation have been described, conclusions about likely behavior can still be reached.

We will be concerned with qualitative methods for inferring behavior in designed situations. Our long-term interest is the inference of behavior in general physical situations, but to simplify the problem somewhat, we will restrict our focus to artifacts (devices) and situations which are arranged to achieve (or not quite achieve) interesting behavior. The intent is to first discover theories which handle simpler situations, later extending successful theories to more general situations.

One approach to this problem is *qualitative simulation* (abbreviated QS from now on). Like simulation in general, a description of the situation is used to determine the relevant parameters (or quantities) and constraints of the situation, a simulation is performed, and the results are transformed into interpretations of the overall behavior. Unlike quantitative simulation, specific values are not assigned to quantities, but only their ordinal relationship to important constants or other quantities are stated. Also, constraints are qualitatively stated. e.g., proportionality may be asserted, but not a specific function. QS then tracks the situation from one qualitative state to another by predicting the changes in the ordinal relationships of the quantities.

We have proposed a different approach called *consolidation* [1], which is a type of qualitative analysis. The behavior of the situation is discovered by inferring the behavior of selected substructures of the situation from the behavior and structure of their constituents. Successive application of this process on increasingly larger substructures results in inferring the overall behavior of the situation.

It would appear, then, that QS and consolidation are rival methods for the same problem. We shall show, though, that they apply to two different problems of behavior inference. The commonalities of the two problems leads to interesting implications about the role of consolidation within a complete theory of reasoning about behavior. Most of the implications described here concern certain difficulties in QS, and how consolidation can be used to handle them. An additional implication pertains to the behavior representation that is used if both QS and consolidation are integrated within a commonsense reasoner.

Our discussion will be divided into three parts. First, we will summarize two quite different approaches to QS, namely those of Forbus [4] and de Kleer and Brown [3], and also outline our consolidation approach. Second, we will identify the different problems that QS and consolidation attempt to solve. Third, we will examine the implications that result from the problems that they work on and the methods that are proposed for solving them.

2. Different Approaches to Inferring Behavior

In addition to summarizing the basic ideas and methods of each approach, we will show how they apply to the example situation pictured in figure 1. In this situation, a flame is under a pan which holds some water. Both the flame and the pan are located in a room.

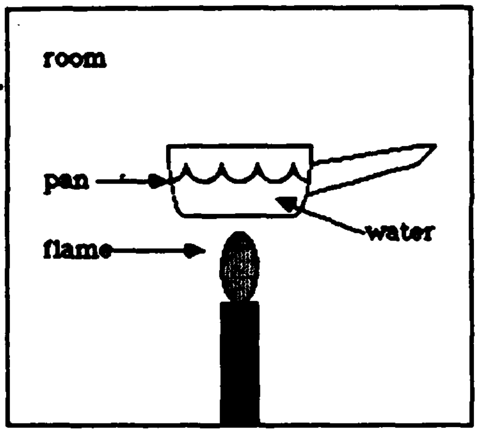


Figure 1: Example Situation

We will talk a great deal about quantities, so a description of them is in order. Quantities are used to represent the real-valued parameters of the QS. So at a specific point of time in a specific situation, a quantity within that situation has a particular real value. For qualitative reasoning, though, always assigning a quantity an actual number is forbidden. Instead important numbers and ranges of numbers are identified as relevant to the quantity (Forbus calls these sets of numbers and ranges *quantity spaces*), and the quantity's relationship within the quantity space is its "value." In addition, the quantity's direction of change (up, constant, or down), i.e., its qualitative "derivative," is maintained for the purposes of the QS, in order to anticipate what its next value will be.

Each of the following descriptions is necessarily too brief to completely describe each approach, so much simplification has taken place. However, they should be accurate enough for the purposes of this paper.

2.1. The Confluence Approach of de Kleer & Brown

This approach models behavior using *confluences*. Roughly, confluences are qualitative equations involving quantities and their derivatives. For example, the following confluence:

$$X + Y = 0$$

indicates a constraint on the *signs* of the quantities X and Y. For example, if X is positive, then Y cannot be positive or zero because then X + Y would be positive, thus Y must be negative. Confluences may also be applied to derivatives of quantities (∂X denotes the derivative of X), so one can specify how quantities move up or down in relation to other quantities. Confluences may refer to any number of quantities or derivatives, and while it is preferred that confluences use only simple addition or subtraction, other operations are allowed.

No agent can be expected to have the set of confluences for each situation that it will experience, so there is a need to describe the structure of a situation, and the behavior of each part of the structure. For de Kleer and Brown, the elements of a situation map into disjoint *components*, which are related to one another via *connections*. Each component is modeled by a set of quantities, and a set of *qualitative states*. Each state specifies when it is active, and a set of confluences which hold when the component is in that state, i.e., the way the component behaves in that qualitative state. Confluences and conditions on qualitative states only reference the components' quantities.

The connections indicate where material is permitted to flow from one component to another. The components of a connection specify which of their quantities are associated with the connection, and the connections are used to determine additional confluences which constrain these quantities. These confluences are used to enforce qualitative versions of general conservation laws, and provide the only means for interaction between components.

The QS is done by a method called *envisionment*, which is a combination of constraint propagation and constraint satisfaction. It is important to note two aspects of envisionment, one concerning the prediction of a temporal sequence of events, and the other with the production of a causal explanation for the values of quantities at each moment of time. For predicting the sequence of events, just constraint satisfaction will do, i.e., begin by determining values for all the quantities which satisfy the confluences, determine which quantity or quantities will next deviate from its current qualitative value, and repeat, solving the confluences (which may have changed because of a change in qualitative state) for the new values.*

Envisionment, however, does not simply satisfy the confluences. Instead, an input disturbance is selected, and its effects are propagated from component to component. If there is not enough information to determine all the

*This is highly simplified since there can be many possible solutions, and many possible "next deviations."

quantities' values, then an assumption about the value of a quantity is made based on heuristics (which de Kleer and Brown have derived from how people explain behavior), and the propagation continues. The path of the propagation and assumptions is used as a causal explanation of the quantities' values.

For figure 1, the flame, the pan, and the room would be considered the components, and would be connected to one another. The water in the pan is not modeled directly, but by appropriate quantities associated with the behavior description of the pan. Heat and temperature are also not directly modeled, but each component (in a more complete model than the one below) would have appropriate heat quantities. For simplification, we will not model boiling or the water vapor that escapes from the pan.

Figure 2 is a simple model of this situation. The flame and room have ideal models of unchanging temperatures ($\partial T_x = 0$). The temperature of the pan varies with the amount of heat flow (represented by the pan's confluence). The heat changes with respect to how much heat flows into (or out of) the pan. Each connection specifies that the amount that flows from one component is the opposite of the amount that flows into the other component. The amount of the flow has the same sign as the difference in temperature.[*]

flame - quantities: T_f (temperature),
Q_{f-p} (heat flow from flame to pan)
Q_{f-r} (heat flow from flame to room)
confluences: $\partial T = 0$

pan - quantities: T_p, Q_{p-f}, Q_{p-r}
confluences: $\partial T_p + Q_{p-f} + Q_{p-r} = 0$

room - quantities: T_r, Q_{r-f}, Q_{r-p}
confluences: $\partial T_r = 0$

each connection - confluences: $Q_1 + Q_2 = 0$
$Q_1 = T_1 - T_2$

Figure 2: Example Model Using Confluences

Suppose that the pan and the room initially have the same temperature, which is lower than the flame's temperature. Taking the temperature of the flame as the input disturbance (we can imagine that it has been just turned on), from the confluences of the connections, heat movement from the flame to the room and pan can be inferred (e.g., $T_f - T_p$ is positive, causing Q_{f-p} to be positive, and Q_{p-f} to be negative). Since the room and the pan are the same temperature, there is no heat flow between them ($T_p - T_r$ is zero, causing Q_{p-r} and Q_{r-p} to be zero). Then from the pan's second confluence, the pan's temperature must be increasing (Q_{p-f} is negative, and Q_{p-r} is zero, making ∂T_p positive). At the next moment of time, the pan's temperature will be higher than the room's, thus heat will flow from the pan to the room. It is now unclear how long the pan's temperature will continue to increase, and it appears possible that the pan's temperature may start to decrease (since Q_{p-f} is negative, and Q_{p-r} positive, any sign of ∂T_p satisfies the confluence.) Adding more confluences can resolve this latter difficulty. If and when ∂T_p becomes zero, adding $\partial Q_1 = \partial T_1 - \partial T_2$ to the connections' confluences will predict that both ∂Q_{p-f} and ∂Q_{p-r} become zero. Now adding $\partial^2 T_p + \partial Q_{p-f} + \partial Q_{p-r} = 0$ to the pan's confluences will predict that the pan's temperature will stabilize.

2.2. The Qualitative Process Approach of Forbus

Forbus introduces a notion called *qualitative process* to account for change and explain why it occurs. Qualitative processes (QP) perform a similar function as confluences, i.e., they both specify behavior and interaction, but the way QPs are defined and applied is very different. First, we need to discuss some of the things that QPs refer to.

Situations are made up of *objects*, predicates on objects, and relationships between them. Forbus does not provide a specific set of relationships, leaving it to the implementor to determine what relationships are relevant to the situation. Additional "individuals" and relationships may be asserted by *individual views*, which consists of a set of conditions, and the relationships which follow from them. An individual view is used to "view" a group of objects as Contained-Liquid individual view. As before, objects and individuals are modeled by a set of quantities.

The only kind of behavior description which may be directly associated with an individual is *qualitative proportionality* between two of its quantities. This simply indicates that a change in one quantity will affect the value of the other quantity. The "direction" of the proportionality is important, indicating which variable is the dependent variable of the relationship.

QPs are the mechanism that determines when changes occur. Unlike confluences, a QP is not part of a individual's behavioral description, but is a general rule which indicates the conditions which cause a quantity or quantities to change in a certain direction. The conditions may refer to any number of individuals. Neither a QP nor a qualitative proportionality guarantees that a quantity will actually change in a certain direction since there may be several active QPs or proportionalities which affect the same quantity. The actual change in a quantity will be the sum of the effects on it.

The QS works as follows: find all the individual views and QPs which are active (whose conditions are true); determine the effects inferred by the QPs and indirectly by any proportionalities; determine what the change(s) will be, viz., a quantity or derivative changes to a new value, a a new QP becomes active, or a previous QP becomes inactive; and repeat.[*]

[*]This last confluence is not quite accurate since if both temperatures are positive, nothing can be concluded about heat flow, i.e., positive minus positive is indefinite. We will assume that it means, e.g., that Q_1 is positive when $T_1 - T_2$ is positive

[*]Since many QPs can affect a single quantity, its direction of change may be ambiguous.

For figure 1, the primary QP will be the heat-flow process displayed in figure 3. The Individuals, Preconditions, and QuantityConditions sections specify the conditions for a heat-flow process to be active. A heat-flow process requires two objects which can store heat, and an object called a heat-path which connects them. It also requires that the path be Heat-Aligned (meaning that there is nothing blocking the flow of heat along that path), and that the temperature of "src" ("A" is a function which refers to the amount of a quantity) be greater than the temperature of "dst". The Relations section specifies additional relations that hold while the process is active. In this process, a quantity called "flow-rate" is created which is greater than zero. The Influences section specifies the effects on quantities. In this case, there will be a negative effect on the amount of src's heat, and a positive effect on dst's heat. The amount of this effect is the amount of flow-rate.

```
process heat-flow

Individuals:
    src an object, Has-Quantity(src, heat)
    dst an object, Has-Quantity(dst, heat)
    path a heat-path, Heat-Connection(path, src, dst)

Preconditions:
    Heat-Aligned(path)

QuantityConditions:
    A[temperature(src)] > A[temperature(dst)]

Relations:
    Let flow-rate be a quantity
    A[flow-rate] > ZERO

Influences:
    I-(heat(src), A[flow-rate])
    I+(heat(dst), A[flow-rate])
```

Figure 3: The Heat-Flow Qualitative Process

The situation in figure 1 can be modeled with the flame, the room, the pan, and the water as objects with heat-paths between the flame, room, and pan. The flame, room, and water each has quantities of heat and temperature. Again, assume that the temperature of the room and the flame remains constant, the flame is hotter than the room, and the room and water are initially the same temperature. Also, the temperature of the water is proportional to the amount of its heat. We will assume that heat-paths to the water are inferred from the Contained-Liquid individual view, or something similar.

Initially, two heat-flow processes are active, from the flame to the room and from the flame to the water. The amount of the water's heat will increase, which because of the proportionality, implies that the water's temperature will increase. Since the temperature of the water and room are now different, another heat-flow process from the water to the room becomes active. Now the same problems as before reappear. It is questionable how long the water's temperature will continue to increase (one heat-flow process has an increasing effect, and the other has a decreasing effect), or even decrease at a later point in time. To avoid the latter problem, the heat-flow process needs to be modified so that the flow-rate is proportional to the temperature difference, and that the flow rate approaches zero as the temperature difference approaches zero. With this modification, if and when the derivative of the water's temperature becomes zero, then the derivatives of all the flow-rates will become zero, and the situation will stabilize.*

2.3. The Consolidation Approach

The consolidation approach uses a structural model similar to de Kleer and Brown's, adding an additional structural relationship called *containment*. Things like water and heat are then modeled as *substances* which are contained by the *components* of the situation, or perhaps other substances, e.g., water contains heat.

Behavior is modeled by specifying the actions (themselves called behaviors) that are performed on substances. Possible actions include:

* Allow. Permits movement of a substance from one place to another
* Pump. Attempts to move a substance through a path.
* Expel. Attempts to move a substance from (or to) a container.
* Move. A substance moves from one place to another.
* Create. Creates a substance within a container.
* Destroy. Destroys a substance within a container.

Each action specifies the kind of substance that is affected, and the location(s) (containers and connections) where it takes place. Each behavior may have a number of parameters or quantities whose values may be real, but is not restricted to be so, e.g., the "rate" of a move signal behavior might be "on" or "off." A quantity may refer to parameters of behaviors that need be inferred, indicating that the behavior is dependent on other behaviors, e.g., the amount of the create light behavior of a light bulb is dependent on the rate of a to-be-inferred move electricity behavior through the light bulb.

A component is modeled by specifying its structure, i.e., its containers and potential connections, and the behaviors which take place within that structure (loosely termed the component's behaviors). A component may have several *behavioral states*, each of which are associated with a different set of behaviors. Each state indicates the condition (a predicate on behaviors) in which the state is active.

Consolidation gets its name from the processing that this

*It is not clear whether Forbus's system can currently perform this analysis, but it is easy to imagine that it can be modified to do so.

approach proposes. The major processing sequence is to instantiate a composite component consisting of a selected subset of components (currently restricted to just two components), and then to infer the behavior of the composite from the behavior of the components. Performing consolidation on increasingly larger composite components results in inferring the behavior of the whole situation. As a byproduct, a hierarchical behavior structure is produced which explains how the overall behavior is caused by the components' behavior.

Behaviors of composite components are hypothesized based on *causal patterns* of behavior and structure. Their existence is confirmed, and their parameters are determined using knowledge about the physics of the substances being acted upon. Roughly then, causal patterns are used to infer behaviors which arise from combinations of other behaviors, and once a pattern is matched, substance-specific knowledge is called upon to fill in the details. Figure 4 illustrates several causal patterns. For example, the serial allow pattern states that if two allow behaviors are in series, then infer an allow behavior over the whole path.

```
Serial Allow:  Allow S from A to B
               Allow S from B to C
               ===> Allow S from A to C

Serial Pump:   Pump S from A to B
               Pump S from B to C
               ===> Pump S from A to C

Expel Move:    Expel S from A
               Expel S from B
               Allow S from A to B
               ===> Move S from A to B
```

Figure 4: Example Causal Patterns

The components of the figure 1 would be the flame, pan, and room, which would all be connected to one another. The pan contains water which contains heat. Both the flame and the room contain heat. Each component has an expel heat behavior whose amount quantity corresponds to the component's temperature. Each component also has allow heat behaviors from its heat container to its connections, and vice versa. The amount of the pan's expel heat behavior is dependent on the amount of heat in the water. The amounts of the other two expel heat behaviors are constant.

Suppose that the pan and flame are initially considered for consolidation. Using the serial allow causal pattern, an allow heat behavior between the pan and the flame is inferred from their individual allow heat behaviors. Using the expel move causal pattern, a move heat behavior between the flame and the pan is inferred from the two expel heat behaviors and the just inferred allow heat behavior. The rate of the move heat behavior corresponds to the temperature difference between the flame and pan, i.e., the difference in the amounts of the expel heat behaviors. Combining the flame-pan composite with the room results in similar use of the serial allow and expel move patterns. The result of consolidation is that there are move heat behaviors between all of the heat containers, and their rates depend on the temperature differences between them.

3. Differences in Information Processing

Consolidation, unlike either QS approach, does not produce a sequence of events as its *output*, yet all three approaches claim to derive the behavior of a situation. Each approach starts from similar models of the situation, and makes inferences about behavior, so how can the final result, conclusions about the situation's behavior, be different? The not-so-earthshattering answer is that consolidation provides a different sort of conclusion about behavior then QS does. QS and consolidation solve two different problems.

3.1. Two Types of Behavior

Part of the confusion comes from the fact that "behavior" is an ambiguous word, and that QS and consolidation pinpoint two of its meanings. The behavior that QS outputs is a temporal sequence of events or states that are predicted to occur in the physical situation. Starting from some knowledge about initial state of the situation, QS attempts to determine a qualitatively complete description of the quantities in the initial state. Using this description, it predicts what the next change (quantities or derivatives changing qualitative value) will be, and determines what the next state will be based on the previous state and the predicted change. By repeating this process, QS predicts the sequences of events that the situation goes though.*

Consolidation outputs what we will call the *potential behavior* of the situation. For example, the move heat behavior between the flame and the pan does not specifically assert when or if heat moves, but that the situation is ripe for heat movement to occur, and that the rate of heat movement can be calculated if some other facts are known, in this case, the temperatures of the flame and pan. The behavior is an indication of what potentially may happen, and points to other information on which this potential is dependent.

3.2. The Information Processing Tasks

The information processing task of a problem is a functional specification of the problem in information terms, i.e., the information that the input and output represent. The previous paragraphs have already identified what the outputs of QS and consolidation are, so here we will attempt to characterize the inputs.

All the approaches require a structural model of the situation as part of their input. The careful reader will have noticed that all the approaches also require some description of how the elements of a situation will behave. In de Kleer and Brown's approach, each element (component and connection) has a behavioral model. In Forbus's approach, each element is modeled by a set of quantities and relations and the interactions among

*It must be remembered that the sequence of events is a qualitative description, so it represents a wide range of actual sequences. Still, the form of the output is temporal and event-based.

elements are described by QPs. In the consolidation approach, each element has a behavioral model, but some general knowledge about the behavior of substances is kept elsewhere.

We have already noted that "behavior" is an ambiguous word with at least two meanings. Does each approach require a new meaning of behavior to describe its input? Our answer is that the input behavior (the behavioral description that is input) of each approach corresponds to the sense of *potential behavior* as described above.

Our argument is as follows. None of the approaches model input behavior as a sequence of events, so that meaning of behavior doesn't apply. The input and output behavior in the consolidation approach are couched in the same representation language, so the input behavior is of the same type as the output. For the QS approaches, the quantities of the situation's elements and their corresponding quantity spaces specify what may potentially happen. Dependencies between quantities are represented by confluences and qualitative states in de Kleer and Brown's approach, and by proportionalities and QPs in Forbus's approach.

Thus the information processing task of QS is:

structure of situation + potential behavior of elements
==> sequence of events

For consolidation, the information processing task is:

structure of situation + potential behavior of elements
==> potential behavior of situation

3.3. Understanding Physical Behavior

What does it mean to understand the behavior of a situation? The two tasks have different views, and would appear to argue against each other as follows. The QS task would say that understanding behavior means being able to determine temporal relations between events. A model of behavior is useless unless it can be used to predict or explain what happens.

The consolidation task would grant that determining the events is important, but would note that QS works because the elements of a situation are well understood, i.e., QS is given a model of their potential behavior. However, QS doesn't provide a true understanding of the situation because it doesn't produce a *like model* of the situation's potential behavior.

Both sides of this debate miss a plausible compromise. Neither task represents a complete understanding of physical behavior, e.g., neither task takes on the problem of designing devices. Understanding, then, does not consist of being able to solve a single information processing task, but in applying a range of problem solving abilities to complex problems. Both QS and consolidation can be viewed as different modalities of understanding behavior. For some problems, QS will be the primary modality, while in others, consolidation will be, while still yet in others, both QS and consolidation will be secondary, perhaps not even needed at all.

4. Implications

Most of the implications in this section identify areas where consolidation can play an important role in reasoning about behavior. Since consolidation takes the same input as QS, it directly competes with QS for certain kinds of reasoning problems. However, to determine what the sequence of events are, QS of some kind is definitely needed, but as argued below, consolidation can still be used to simplify the work of QS.

4.1. Initial Conditions

Suppose that in figure 1, no initial conditions (initial values of quantities) were known, but some statement about the situation's behavior is still desired. Without initial conditions, QS is unable to start. The best that could be done would be to enumerate all the possible initial conditions and perform QS on each possibility. An enumeration of initial states might be small in this simple case, but in more complex situations, there would be many possible initial states.

Consolidation can proceed without assuming any initial conditions, and in fact, the processing described earlier did not do so. If we examine more closely what the final result looked like (figure 5), it is not hard to see why this is the case. Each quantity is defined not in terms of specific values at specific moments of time, but in terms of how it is dependent on other quantities. Thus if the pan happened to be hotter than the flame, then the rate of heat flow from the flame to the pan would be negative, indicating that heat would flow from the pan to the flame. Figure 5 is a condensed representation of potential behavior which can be directly used to answer questions and for other purposes, including QS.

```
rate[move heat from flame to pan]
    = proportional( amount[expel heat from flame]
                    - amount[expel heat from pan])

rate[move heat from flame to room]
    = proportional( amount[expel heat from flame]
                    - amount[expel heat from room])

rate[move heat from pan to room]
    = proportional( amount[expel heat from pan]
                    - amount[expel heat from room])

amount[expel heat from pan]
    = proportional( amount[heat within pan])

amount[expel heat from flame] = constant

amount[expel heat from room] = constant
```

Figure 5: Results of Consolidation

4.2. Open Systems

A similar problem for QS is deriving the behavior of situations which are open systems, i.e., there is interaction between the situation and the outside world. Without knowledge of what these interactions are, the value of each quantity that can be affected becomes indeterminable. Enumeration of all conceivable outside interactions is not, in general, a feasible solution since the number, kind, and order of interactions can vary greatly. However, the ability to reason about open systems seems to be necessary for understanding behavior since most situations that an agent could be expected to encounter are open systems, and parts of situations are by definition open systems.

By providing a condensed *representation* of potential behavior, consolidation gives a solution to describing the behavior of open systems. If we changed the model of the room in our example so its temperature could fluctuate, and it had a potential "heat connection" to the outside, the result of consolidation would not be fundamentally changed. The only difference is that the room could gain or lose heat through other interactions. Heat will move among the room, flame, and pan in pretty much the same way.

4.3. Causal Explanation

Two types of causal explanation correspond to the two types of behavior defined earlier. QS emphasizes the causality of temporality and propagation, i.e., the current state of the situation leads to the next, and the value of one quantity changes (via some confluence or QP) the value of another quantity. Consolidation emphasizes the causality of composition, i.e., the behavior of a group of components arises from the behavior and structure of the individual components. Another debate like the device understanding debate could be promulgated at this point with probably the same result. Neither type of causality is necessarily superior to the other, but their usefulness depends on the particular problem to be solved. It is worthwhile to note that there is obvious causality in situations with unknown initial conditions and in open systems (consider two batteries connected in series). Consolidation can be used to point out this aspect of causality.

4.4. The Complexity of Qualitative Simulation

QS is a global reasoning process. To perform the simulation for a particular moment in time and to check if it has been done consistently, all the elements of the situation must be taken into account. For example, the derivative of every quantity must be examined to update the quantities' values. This is true no matter the number of quantities and confluences (or active QPs) the situation model has. The nature of QS prevents a hierarchical breakdown since any part of a situation is very likely to be an open system, and also because the information processing task of QS is not recursive. The output is not the same kind of information as the input.

Integrating consolidation with QS would help alleviate this difficulty. Consolidation can be used to determine a potential behavior description of the situation or disjoint parts of it, and QS can then be applied to the reduced situation. In other words, even if a sequence of events is the desired output, consolidation can be used to reduce the apparent complexity of QS.

4.5. Diagnosis

The introduction suggested that robust domain models would lead to better diagnostic reasoning. Some recent research is based on diagnosing from a representation of the structure and function of the domain [5, 2, 6]. While this research varies on a number of details (e.g., both Genesereth's and Davis's systems perform diagnosis based on their representations, while Sembugamoorthy and Chandrasekaran generate a diagnostic system from theirs), they have the common feature of hierarchical representation of structure and behavior. Consolidation is potentially applicable for generating, explaining, and verifying behavioral descriptions at each level of the hierarchy. *Since qualitative simulation is not suited to* hierarchical descriptions, it is hard to see how it can applied.

The behavioral structure that consolidation produces could also be used for diagnostic reasoning. There are two possibilities. The first is if an inferred behavior does not occur when predicted. In this case, there must be some change in what was supposed to cause the behavior to occur. The second, more difficult case, is if some some behavior occurs when it is not predicted. Since this behavior must have been caused, every causal pattern which can imply this behavior becomes a potential hypothesis. The possible ways that this pattern can be satisfied are subhypotheses. For example, if substantial heat is unexpectedly moving from one place to another, one possibility is that the expel move pattern was satisfied. This pattern (with knowledge about heat) requires that the two places have different temperatures, and that there is some path between them along which heat can flow. Different possible heat paths constitute different subhypotheses. This kind of reasoning could also serve as the basis for learning how things behave.

4.6. Representation

Both consolidation and QS have the same kind of input, so representations of potential behavior should be amenable to both kinds of problem solving. From the consolidation point of view, representations should facilitate the composibility of behaviors. The representations of the QS approaches do not have this property.

In de Kleer and Brown's representation, consolidation would need to derive the confluences and quantities of composite components from the confluences and quantities of individual components. Simply appending the models of the confluences and their connections together into one description doesn't provide any additional information at all and would eventually make composite components too complex for comprehension, so these confluences need to be simplified. Unfortunately, equation operations like substitution do not apply to confluences. For example, the following deduction is incorrect for confluences.

$$W = Y + Z \text{ and } X = Y + Z \implies W = X$$

If Y is negative and Z is positive, W and X can have different signs without any contradiction, therefore W = X does not follow. This inability to simplify confluences would make it extremely difficult to use consolidation on this representation.

The initial difficulty for consolidation with respect to Forbus's representation is that composing behaviors doesn't make any sense. Individuals do not have behaviors; instead, QPs specify all direct effects that take place. The alternative is to specify composite components so that QPs will correctly apply to them, i.e., giving composite components the right quantities and relationships. Doing this will require something isomorphic to the causal patterns and the substance knowledge that consolidation currently uses. For example, to derive the voltage quantity for two batteries in series, we need to know that the two batteries will have its own voltage quantity (a pump electricity behavior is caused by two pump electricity behaviors in serial), and that in this kind of configuration, voltage is additive (the electrical knowledge that is invoked when the serial pump pattern is satisfied). So Forbus's representation has no special advantages, and would actually obscure the underlying regularity (the serial pump causal pattern). Another difficulty is when a process occurs inside the composite component, e.g., heat moves within the flame-pan composite.

4.7. Caveats

Currently, consolidation is not able to model all the phenomena that the QS approaches are able to. Also, there may be certain kinds of situations in which consolidation would not work very well.

Forbus's approach is able to handle spatial reasoning that is more complex than the connection and containment variety. For example, he can model, to a limited extent, motion and collision of objects. Forbus can also model properties of material, changes in material, and changes in structure. Consolidation has not been extended to cover these phenomena.

One possible problem for consolidation is constructing a plan for composing components in a sensible way. We are developing general heuristics for making this decision, e.g., select composite components with as few outside connections as possible. Also, domain knowledge could provide additional heuristics. In some cases, this won't be an issue because a structural hierarchy will be imposed by an outside agent.

The analysis that consolidation provides holds only as long as the components remain connected in the same way. If a connection (or a component, for that matter) is created or destroyed, then much of the analysis will be invalid and must be redone. This need for reanalysis would make consolidation poor on situations where the structure frequently changes.

Currently, consolidation is heavily tied to dividing the elements of a situation into components and substances. In some situations, this division cannot be strictly applied. It is usually reasonable to model a light bulb, for example, as a component, but to be able to reason about a ball colliding with the light bulb, the light bulb would need to be modeled as a substance which is contained by the space it is in, and the ball is trying to get into the same "container." Since this will probably affect the light bulb's behavior, the two perspectives need to interact. A change of perspective is also necessary for substances which can viewed in different ways. For example, Davis [2] notes that the bits within a digital circuit also need to be analyzed as electricity in order to understand certain problems such as power failure.

5. Conclusion

In the previous section, we have stressed the merits of consolidation in comparison to qualitative simulation. To understand what we wanted to accomplish by this, the full context of the discussion must be considered. We have carefully, although briefly, summarized three methods that perform qualitative reasoning about physical behavior. We have shown that one of these methods, consolidation, solves a different information processing task than the other two methods, which both perform qualitative simulation. Thus consolidation cannot directly substitute for qualitative simulation. However, all the methods accept the same kind of input, and their output is about behavior, albeit different aspects of behavior. It is possible that this difference is uninteresting, e.g., it might be that wherever consolidation can be used, qualitative simulation can be used to achieve the same effect. Therefore to understand the role of consolidation and the relationships between these tasks, we have shown where consolidation play a major role in qualitative reasoning.

Acknowledgments

I would like to thank B. Chandrasekaran, John Josephson, Jon Sticklen, and Mike Tanner for their discussions, comments, and encouragement. This research is supported by the Air Force Office of Scientific Research through AFOSR grant 82-0255.

References

1. T. Bylander and B. Chandrasekaran. Understanding Behavior Using Consolidation. Proc. Ninth International Joint Conference on Artificial Intelligence, IJCAI, Los Angeles, 1985. To appear.

2. R. Davis. Diagnostic Reasoning Based on Structure and Function. *Artificial Intelligence 24* (1984), 347-410.

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

4. K. D. Forbus. Qualitative Process Theory. *Artificial Intelligence 24* (1984), 85-168.

5. M. R. Genesereth. The Use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence 24* (1984), 411-436.

6. V. Sembugamoorthy and B. Chandrasekaran. Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems. AI Group, Dept. of Computer and Information Science, The Ohio State University, 1984. To appear in *Cognitive Science*.

Appendix F

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

Technical Report

July, 1985

UNDERSTANDING BEHAVIOR USING CONSOLIDATION

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

Presentation at The 9th International Joint Conference on Artificial Intelligence, University of California, Los Angeles, California, Aug 18-24, 1985. To appear in IJCAI-85.

Understanding Behavior Using Consolidation

Tom Bylander and B. Chandrasekaran

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

Abstract

In this paper, we wish to make three contributions to Naive Physics in the context of reasoning about devices. (1) We discuss some limitations of current qualitative simulation approaches with regard to a number of issues in understanding device behavior and point to the need for additional processes. (2) We introduce a new approach to deriving the behavior of devices called consolidation. In this approach, the behavior of a device is derived from the behavior of its components by inferring the behavior of selected substructures of the device. (3) We present an ontology of behavior and structure which is well-suited to the consolidation process. This ontology makes it possible to state rules of behavior composition, i.e., simple patterns of behavior and structure are used to infer additional behaviors.

1. Introduction

Naive Physics is the commonsense knowledge that people have about the world. This knowledge includes the ability to *qualitatively* understand the behavior of physical systems. Our investigation is presently concerned with a subset of physical systems, focusing on designed artifacts or devices. Ultimately, we are interested in developing a representation which is applicable to a wide variety of understanding problems. However, the research described here is conducted specifically on the problem of deriving the behavior of a device given its structural description and the behavior of its components.* We hope to integrate the results of this research with other work concerning the functional representation and diagnosis of complex devices [6].

One recent approach to this problem is *qualitative simulation* [2, 3, 5]. The description of the device determines the relevant quantities and constraints of the simulation, a simulation is performed, and the results are transformed into interpretations of the device's overall behavior.

This differs from quantitative simulation in several ways. Instead of assigning specific values to a quantity, only its ordinal relationship to important constants or other quantities is stated. Constraints are also qualitatively stated, e.g., proportionality may be asserted, but not a specific function. In addition to constraint satisfaction (the analogue of simulation by numerical methods), the techniques of qualitative simulation include constraint propagation, and matching descriptions of potential processes. The process of interpretation extracts state transition information, summarising the possible behaviors and inferring causal relationships between device states.

*Other problems include design, diagnosis, planning (using devices to accomplish a goal), etc.

We propose an alternative approach that is a type of qualitative analysis. The behavior of the device is discovered by inferring the behavior of selected substructures from the behavior and structure of their components. Successful application of this process on increasingly larger substructures results in inferring the behavior of the device. This approach, called *consolidation*, has a number of desirable properties, including localised reasoning steps, causal analysis of behavior, and consistency of representation. Consolidation is not intended to be a complete solution to the inference-of-behavior problem, but where it can applied, we believe that it is a better alternative for analyzing and explaining behavior. We wish to emphasize at this point that an implementation of this approach is currently in progress.

The notion of reducing complexity by reasoning about a group of subcomponents as a single abstract component is shared by the work of Sussman and Steele [7] and is embodied in their notion of "slices." However, the aims and methods of their proposal make the details very different.

First, we argue that qualitative simulation has several undesirable characteristics as a Naive Physics theory. Next, we introduce consolidation, dividing the discussion into the description of components and the inference of behavior. A difficult example is then analysed. Finally, unresolved issues are discussed.

2. Critique of Qualitative Simulation

2.1. Complexity

One desirable property of a Naive Physics theory is *simplicity of computation*. While current theories of qualitative simulation (QS) may be useful for providing upper bounds on the competence of qualitatively reasoning agents, they are unsatisfactory to account for human reasoning behavior due to the following two reasons.

First, QS is a *global reasoning process*. To perform the simulation for a particular point in time and to check if it has been done consistently, all the quantities and constraints must be taken into account. To go from one time point to another, the derivative of every quantity must be examined to update the quantities' values. This is true no matter the number of quantities and constraints the device has. A hierarchical breakdown is difficult because QS relies on nearly-closed systems (boundary conditions must be known or enumerable) and on constraint propagation. Forbus's notion of p-components [3] provides a method for subdividing a situation into independent parts. However, when the parts are more mutually dependent as in a device, additional techniques are called for.

Second, some theories of QS involves *substantial mathematical reasoning*. Quantities and their derivatives must be carefully handled so that constraints are not violated, and continuity is maintained. Since the constraints are stated in terms of arithmetic and differential relationships, constraint propagation and checking for consistency require a considerable amount of mathematical reasoning. It must be stated, however, that Forbus's approach avoids much of the complexity of constraint propagation by restricting the paths over which propagation can occur.

2.2. Causality

Another desirable property is explaining the device's behavior in terms of the behavior and structure of its components; we want to know the *cause* of the device's behavior. Causality in some theories, especially the confluence theory of de Kleer & Brown [2], is identified mainly with the propagation of values through constraints.

The major problem with this position is that causality is viewed as a *"last straw" phenomenon*, i.e., the saying "the last straw broke the camel's back" would be translated, in this view, to "the last straw was the cause of the camel's broken back". De Kleer & Brown admit that their version of QS does not identify "the support which enables the causal action path to exist" [1]. However, it seems wrong to omit the support from a causal account, since the support may include the primary causal processes of the effect (e.g., most of the weight is already on the camel's back).

2.3. Representation

QS theories require descriptions of components to specify their outward structure, the quantities that are involved in interaction with other components, the constraints on those quantities, and the behavioral states.* This description may be thought of as the *behavioral laws* of the component. On the other hand, the representation of device behavior does not describe its behavioral laws, but is a network which shows the temporal (and causal) sequence of the components' states. If this process were to be repeated one more level (i.e., where the device at this level becomes a component at the next level), QS is not helpful, since it needs to have the behavioral laws of the device. This twin representation of behavior together with the global nature of QS limits the applicability of this approach.

Another problem with current representations is the *ontological impoverishment* of a theory primarily based on quantities and constraints. It is the burden of the model-builder to insure that the right types of quantities and constraints are represented and consistently defined. While there are guidelines for how to do this, these guidelines are *outside the representational system*. For example, Ohm's law is very significant for describing the behavior of electrical components. However, Ohm's law itself is not represented *in* QS, but *is compiled into each component* description that depends on it.**

3. Consolidation: Description of Components

Components interact with other components. The interaction is not just about components, but about the "stuff" or substances which potentially move between components and affect their behavior. What does a component have so that interactions can occur? We believe that a commonsense answer has two parts. One thing a component has is *structure*. On its exterior, it has places which are used to connect it to other components. On its interior, it has places which hold or contain substances. The other thing a component has is *behavior*, how it acts and is acted upon by substances. This section discusses how we represent the structure and behavior of components; the following section describes what inferences this representation supports.

3.1. Structural Primitives

Like de Kleer & Brown, we will use *connection* to signify that one component is attached to another component or is otherwise in meaningful spatial contact with it. An example of "meaningful spatial contact" is the relationship of the surface of a light bulb with the space around it, which in turn, might be in contact with something that the light affects. Note that we also include empty space as a type of component. This is essential for reasoning about movement though space, and about magnetism and gravity.

We also use *containment* as a structural relationship to represent the places inside components that substances can move from, move into, and be at rest. These places may or may not have significant capacity. The importance of this concept for Naive Physics theories was emphasized by Hayes [4].

For example, the light bulb in figure 1 has three connections: two electricity connections called "end1" and "end2", and a light connection called "surface". Inside of the light bulb, there are places where electricity passes through, and where light is produced. To model this, containers called "electrical" and "light source" are attributed to the light bulb, and are used in the behavioral description on the right.

3.2. Types of Component Behavior

Components act upon substances. We propose to describe these actions by a small set of relationships, using them as a foundation for representing additional knowledge about components and substances. They are:

- *Allow*. The component permits a specified kind of substance to move from one place to another.
- *Influence*. The component tries to move a specified kind of substance. There are two subtypes according to the spatial relationship of the **influence** with potential sinks and sources.
    - *Pump*. The component tries to move a substance through it, e.g., a battery has a **pump** electricity behavior from the negative to the positive terminal. The sink and source are external to a **pump** behavior.
    - *Expel*. The component tries to move a substance from (or to) an internal container, e.g., a balloon has a **expel** air behavior.
- *Move*. The component moves a specified kind of substance from one container to another along a specified path. Move behaviors are implicitly constrained by the amount and capacity of the containers.
- *Create*. The component creates a specified kind of substance in a container, e.g., a light bulb has a **create** light behavior.
- *Destroy*. The component destroys a specified kind of substance in a container, e.g., an acoustic insulator has a **destroy** sound behavior.

For example, the light bulb in figure 1 has **allow** electricity,

*Different "behavioral states" are associated with different sets of constraints. The total state of a component is its behavioral state and the values of its quantities.

**This criticism doesn't apply to Forbus's *Qualitative Process* theory [3], i.e., a single qualitative process description can be used to represent Ohm's law.

```
    gate|
        |           allow electricity between end1 and end2
    +---------+         thru electrical,
end2|         |        resistance 0, state closed
+---| switch |     allow signal from gate to sensor
|   |         |     change state to closed
|   +---------+         when [move signal from gate
|       |end1                        to sensor, rate on]
|       |
|       |
|  +----------+     pump electricity from negative terminal
|  |    +     |         to positive terminal thru electrical,
|  | battery |          amount positive
|  |    -     |     allow electricity between negative term.
|  +----------+         and positive terminal thru electrical,
|       |               resistance positive
|       |
|       |end2       allow electricity between end1 and end2
|   +--------+          thru electrical, resistance positive
|   |        |      allow light between light source
+---| light |           and surface
end1| bulb   |      create light in light source,
    |        |          rate [proportional (magnitude (rate
    +--------+                (move electricity between end1
        |                          and end2)))]
 surface|
```

Figure 1: Light Bulb Device

allow light, and create light behaviors (for the purposes of this discussion, other behaviors of the light bulb and other components have not been displayed). There are conditions on some of these behaviors, which are specified in the details of the description. For example, the **create** light behavior is dependent on movement of electricity. There is more discussion on this later.

Some components, such as the switch, have different behavioral states, where each state is associated with different behaviors. An additional type of behavior, *change state*, specifies a predicate on behavior and the next state of the component. For example, the switch in figure 1 has two states, open and closed, where the closed state has an **allow** electricity behavior, and the open state does not. The switch also has an **allow** signal behavior, and it will **change** state depending on the control signal that it receives.

3.3. Quantities

We use quantities to describe additional detail about behaviors and containers. Most of the behaviors have a natural measurement: **move** by rate of movement, **create** by rate of creation, **destroy** by rate of destruction, and **influence** by amount of influence. Also, some behaviors, especially **allow** behaviors, may have special quantities which are specific to the substance. Resistance, capacitance, and inductance are examples from electricity. The **allow** electricity behavior of the light bulb, for example, has a positive resistance.

Each container has quantities which describe its capacity and amount. The containers of the components in figure 1 can be modeled with infinitesimal capacity, so interesting issues concerning these quantities do not arise. In section 5, we will discuss an example in which these quantities have significant behavioral consequences.

Quantities can be used to express how some behaviors are dependent on other behaviors, i.e., how the component is acted upon by substances. For example, the rate quantity of the **create** light behavior of the light bulb is described as proportional to the rate of a **move** electricity behavior which goes through the light bulb.

4. Consolidation: Inference of Behavior

We propose to infer the behavior of a device by a form of composition. The behavior of selected substructures or *composite components* of the device is inferred from the behavior and structure of their subcomponents. Composites are used as *contexts* for forming intermediate points of understanding about the device. This composition is possible because the behaviors of components as represented above are themselves composable; certain behavioral and structural patterns give rise to additional behaviors. These *causal patterns* are also used to index into knowledge about the behavior of substances, i.e., knowledge about substances is organized around the possible generic situations in which behaviors are inferred.

4.1. Causal Patterns of Behavior and Structure

A causal pattern describes a situation in which a behavior may occur, asserting that if certain behaviors satisfy a specific structural relationship, then another behavior of a specified type may be caused.[*] For example, the **propagate pump** pattern specifies that a **pump** behavior in a serial relationship with an **allow** behavior will potentially cause another **pump** behavior, e.g., a **pump** electricity behavior between A and B, and an **allow** electricity behavior between B and C may cause a **pump** behavior between A and C. Whether this **pump** behavior actually occurs depends on the physics of the substance and the details of the sub-behaviors. The following are the causal patterns that we have discovered so far:

- *Serial/parallel* **allow**. An **allow** behavior caused by two serial or parallel[**] **allow** behaviors.
- *Parallel* **pump**. A **pump** behavior caused by two **pump** behaviors in parallel.
- *Propagate* **pump**. A **pump** behavior caused by a **pump** and an **allow** behavior in serial.
- *Propagate* **expel**. An **expel** behavior caused by an **expel** behavior and **allow** behavior in serial.
- *Serial/parallel* **move**. A **move** behavior caused by two serial or parallel **move** behaviors.
- **Pump move**. A **move** behavior caused by a **pump** behavior and an **allow** behavior, both on the same path from one container to another. In this pattern, the source and sink may be the same container in which case the movement is around a circuit.
- **Expel move**. A **move** behavior caused by an **allow** behavior which "connects" an **expel** behavior to another container.

We do not claim that this list is complete. Additional patterns may be required to reason about concepts like momentum, in which movement leads to additional influences. However, we believe that the number of additional patterns will be small.

[*] Currently, our theory does not handle situations in which the behaviors satisfying a pattern refer to different substances, e.g., oil and water.

[**] Roughly, two behaviors are "serial" if they share an endpoint; two behaviors are "parallel" if they have the same endpoints.

Suppose that a composite of the battery and the switch in figure 1 is chosen for processing. Behaviors are inferred based on the causal patterns as follows:

- Using the **serial allow** pattern, an **allow** behavior between the negative terminal of the battery and end2 of the switch is inferred. The resistance is determined to be positive from knowledge about electricity. Since the switch's **allow** behavior *is active only during the closed state, the same is* true of the inferred behavior. Since the states of the switch result in different behavior of the battery–switch, the battery–switch also has open and closed states.
- Using the **propagate pump** pattern, a **pump** behavior from the negative terminal of the battery to end2 of the switch is inferred. The amount is determined to be positive.

The causal patterns do not take into account that the battery–switch will also have the **allow** signal and **change state** behaviors of the switch. In general, those behaviors which affect the outward behavior of the composite, and which are not subsumed by an inferred behavior need to be copied to the composite. Also, note that none of the causal patterns refer to **create** and **destroy** behaviors. These kinds of behaviors are transferred to the composite if they are connected to the "*outside*" by **allow** behaviors.

The causal patterns are similar to Forbus's process descriptions [3]. Both describe the conditions necessary for some behavior to happen. One important difference is that the causal patterns are generic to all substances. While a process description can be stated at a high level of generality, there is no commitment by the theory to any particular level of generality. In practice, there are different process descriptions for different *types of substances such as liquid, gas, heat, etc. Also, the* process descriptions can be used only when changes occur, while the causal patterns can handle situations, such as two batteries connected serially, in which no physical change takes place.

4.2. Simplification of Structure

If a composite simply inherited the structure of its subcomponents, the description of larger composites would become increasingly complex, making it harder to reason about them. This is allayed in two ways. First of all, only the external connections of the composite become part of its behavioral description. For example, the positive terminal of the battery and end1 of the switch would not be referenced in the battery–switch's description.*

Second, *composite containers* may be instantiated as combinations of several other containers. In the battery–switch, the electrical containers of the battery and switch are combined to form a single electrical container. The creation of composite containers is governed by the inference of behaviors, under a constraint that restricts behaviors to reference only a limited number of connections and containers. For example, the "thru" attribute of the inferred **allow** electricity behavior of the battery–switch may only reference one container, thus a composite container is instantiated.

4.3. Physics of Substances

The physics knowledge contains the procedures that are used to validate inferred behaviors and determine the values of their quantities. Each substance has procedures which are associated with the causal patterns, and with other known situations such as dependencies and the inference of composite containers. For example, when the **serial allow** pattern matches on behaviors involving electricity, the resistance of the caused behavior is determined by summing the resistances of the causing behaviors.

The reasoning is more complicated when dependencies are involved. Suppose that we chose a composite consisting of the light bulb and switch. This composite will also have a **create** light behavior, which should have a rate quantity specified as:

```
proportional (magnitude (rate
    (move electricity between end1 of the switch
        and end2 of the light bulb)))
```

The places mentioned by the dependency must be part of the simplified structure of the composite. To do this, there must be knowledge of what paths through the composite will also go through the light bulb, and the dependency must be modified accordingly.

4.4. Light Inference

The primary effect of the light bulb system is that light is produced when the switch is closed. Consider now a composite which consists of the battery–switch and the light bulb. This inference can proceed as follows:

- The **allow** electricity behaviors of the battery–switch and light bulb satisfy the **serial allow** pattern, resulting in an **allow** *electricity behavior around the electrical circuit.* The resistance is positive. The behavior is active only during the closed state.
- The **pump** electricity behavior of the battery–switch and the **allow** electricity behavior of the light bulb satisfy the **propagate pump** pattern, from which a **pump** electricity behavior around the circuit is inferred. The amount of the behavior is positive. The behavior is active only during the closed state.
- The two behaviors inferred above satisfy the **pump move** pattern, so a **move** behavior around the circuit is inferred. The rate of the move is positive. The direction depends on how electricity is modeled.
- This **move** behavior satisfies the dependency expressed in the **create** light behavior of the light bulb. The rate of creation is calculated as positive.

In the inference of the rate of creation, every behavior of the components and element of structure which plays some role in the creation of light has been used in the consolidation process. *The explanation of this inference provides a complete causal account of the creation of light in the light bulb system in terms of the components' behavior and the device's structure.*

Also note that all the electrical connections are internal to the device. Thus no electricity behavior becomes part of the final description of the device's behavior. *The device's behavioral description states only what the outward behavior of the device is, not how it is accomplished.*

5. Another example

To further illustrate how consolidation works and to explain additional features of this analysis, consider the situation in figure 2. The source and sink components have containers of water of differing temperatures. The source component has an expel water behavior. There is a connection between the components which permits the flow of water.

*Connections which connect two or more components are assumed to be internal to the device, unless declared otherwise.

```
+---------+
|         |  container a of water, temperature x
| source  |  expel water from a
|         |  allow water between a and hole
+---------+
   |hole
   |
   |hole
+-------+
|       |  container b of water, temperature y
| sink  |  allow water between b and hole
|       |
+-------+
```

Figure 2: Water Containers

A difficulty in modeling this device is representing temperature. We will say that the water within the a and b containers *contain* heat, and that water has a expel heat behavior. The amount of the expel behavior corresponds to the water's temperature. This extended notion of containment can also be used to model concentrations of dissolved material.

When water movement is inferred, heat movement should also be inferred. The movement of heat will not only affect the amount of heat, but will affect the amount of the expel heat behavior within the containers. These inferences are general enough to be codified as causal patterns:

- A move behavior of a substance S1 which contains a substance S2 causes a move S2 behavior along the same path.
- A move behavior of a substance S1 which contains a substance S2 and which has an expel S2 behavior affects the expel S2 behavior at the sink of the move.

For heat, the effect is that the expel heat amount of the sink will tend towards the expel heat amount of the source, i.e., the temperature of the sink will move towards the temperature of the source.

With these additional patterns, the inference of the move water behavior will lead to inferring a move heat behavior, and to consideration of the effect upon the expel heat behavior of the sink. In this case, the capacities of and amounts within the two water containers will affect what will actually happen. When initial values for these are chosen, the behavioral description can then be run to determine what sequence of events will occur.

6. Summary

In Section 2, we discussed several problems with qualitative simulation in understanding device behavior. Here, we discuss how consolidation overcomes some of those difficulties.

In consolidation, the reasoning occurs in well defined, locally contained steps. Mathematical ability determines the sophistication of analysis, but is not required to perform it. To claim simplicity, an implementation is necessary to demonstrate that the analysis can be done efficiently in real situations. The primary complication is deciding what composites should be analysed. We are developing general heuristics for making this decision. Also, domain knowledge could provide additional heuristics, such as a library of precompiled composites that are instantiated when they are recognised.

Causality is directly linked to the idea that the components of the device cause the behavior of the device. Reasoning about behaviors using the causal patterns leads to an inference structure showing the behavioral relationships between the components and the device. A complete account includes the dependencies between behaviors, and how they are satisfied.

The representation distinguishes between different types of quantities and constraints. Different quantities are associated with the behaviors and substances that they describe. Constraints are embedded within dependencies and the physics knowledge of substances. The device's behavior is represented in the same manner as a component's behavior.

A possible problem is that states of components almost always become state of composites. A combinatorial problem might occur when several components have multiple states. One alternative is to use simulation in this kind of situation. A more interesting alternative is to simplify the description of composites using various means such as inferring that certain states are impossible, combining related states into a single state, etc.

Consolidation is but one of the multiplicity of processes and representations that are a part of Naive Physics reasoning. Further research is called for in describing the relationship between consolidation and qualitative simulation, in expanding the richness of the structural primitives, and in representating and integrating, for example, discreteness of motion, temporality, and mixing of substances.

Acknowledgments

We would like to acknowledge John Josephson, Jon Sticklen, and Ken Forbus for their comments on earlier drafts of this paper. Of course, this doesn't imply endorsement of the final product. This research is funded by the Air Force Office of Scientific Research through AFOSR grant 82-0255.

References

1. de Kleer, J., and Brown, J. S. Assumptions and Ambiguities in Mechanistic Mental Models. In *Mental Models*, Lawrence Erlbaum, Hillsdale, New Jersey, 1983, pp. 155-190.

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

3. Forbus, K. D. Qualitative Process Theory. *Artificial Intelligence 24* (1984), 85-168.

4. Hayes, P. J. Naive Physics I: Ontology for Liquids. 35, Institut pour les Etudes Semantiques et Cognitives, Unversite de Geneve, Geneva, 1978.

5. Kuipers, B. Commonsense Reasoning about Causality: Deriving Behavior from Structure. *Artificial Intelligence 24* (1984), 169-203.

6. Sembugamoorthy, V., and Chandrasekaran, B. Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems. Technical Paper, AI Group, Dept. of Computer and Information Science, The Ohio State University, 1984. To appear in *Cognitive Science*.

7. Sussman, G. J., and Steele, Jr., G. L. CONSTRAINTS -- A Language for Expressing Almost-Hierarchical Descriptions. *Artificial Intelligence 14* (1980), 1-39.

Appendix G

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

Technical Report

July, 1985

MAPPING MEDICAL KNOWLEDGE INTO CONCEPTUAL STRUCTURES

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

Jack W. Smith, Jr., M.D.
Laboratory for Knowledge-Based Medical Systems
Department of Pathology
The Ohio State University
Columbus, Ohio 43210

Invited presentation at The Expert Systems in Government Symposium, McLean, Virgina, October 24-25, 1985. To appear in the Proceedings of the Conference.

Mapping Medical Knowledge into Conceptual Structures

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

Jack W. Smith, Jr., M.D.
Laboratory for Knowledge-Based Medical Systems
Department of Pathology
The Ohio State University
Columbus, Ohio 43210

Abstract

CSRL (Conceptual Structures Representation Language) is a language for implementing the classification portion of an expert diagnostic system. Our approach to classification calls for a problem solving structure that is organized as a classification hierarchy (e.g, a classification of diseases). Each hypothesis in the hierarchy is associated with a "specialist" which performs the decision-making activity for the hypothesis. A top-down strategy called establish-refine is used in which either a specialist establishes and then refines itself, or the specialist rejects itself, pruning all the specialists below it. CSRL is a language for representing the specialists of a hierarchy, and the knowledge embedded within them. This paper discusses how medical knowledge should be applied to the two critical design problems of CSRL: forming a hierarchy which is compatible with the establish-refine strategy, and implementing specialists that accurately evaluate the plausibility of their corresponding hypotheses.

1. Introduction

CSRL (Conceptual Structures Representation Language) is a language for implementing classification problem solvers. Our approach to classification is an outgrowth of our group's experience with MDX, a medical diagnostic program [6], and with applying MDX-like problem solving to other medical and non-medical domains. CSRL facilitates the development of diagnostic systems by supporting constructs which represent classificatory knowledge at appropriate levels of abstraction. The intent is to allow the system implementor to more directly encode the knowledge acquired from domain experts, and to avoid much of the detail associated with general purpose languages.

We will focus on how to use CSRL to construct diagnostic expert systems in the medical domain. Our motivation for doing so is because CSRL's intended use, like every other language, cannot be expressed solely by describing its syntax and semantics. For example, how one diagnostic representation should be compared with another, or how the knowledge should reflect how a physician thinks or the way the body functions are questions which are not resolved by the language.

We will discuss how medical knowledge interacts with the constraints of CSRL to influence the design of an expert system. In particular, two critical design problems must be faced:

- forming a classification of hypotheses which is compatible with the basic establish-refine strategy of classification; and
- encoding knowledge for each disease so that the plausibility of the disease is accurately evaluated.

Before we enter this discussion, however, we will introduce our approach to diagnostic problem solving and the main features of CSRL.

2. Introduction to Diagnostic Problem Solving

An important part of diagnosis is the classification of case data into plausible malfunctions. Classification is a specific type of problem solving in our approach, meaning that a special kind of organization and special strategies are strongly associated with it. In this section, we will briefly review the theory of problem solving types as presented by Chandrasekaran [3, 4, 5], the structure and strategies of the classification task, and the role of classification in diagnosis.

2.1. Problem Solving Types

We propose that expert problem solving is composed of a collection of different problem solving abilities. The AI group at Ohio State has been working at identifying well-defined types of problem solving (called generic tasks), one of which is classification. Other examples include knowledge-directed data retrieval, consequence finding, and a restricted form of design.

Each generic task calls for a particular organizational and problem solving structure. Given a specific kind of task to perform, the idea is that specific ways to organize and use knowledge are ideally suited for that task.

Even when the specification of a problem is reduced to a given task within a given domain, the amount of knowledge which is needed can still be enormous (e.g., classifying diseases). In our approach, the knowledge structure for a given task and domain is composed of *specialists*, which correspond to different concepts of the domain, and perform the decision-making activity associated with that concept. Domain knowledge is distributed across the specialists, dividing the problem into more manageable parts, and organizing the knowledge into pieces which become relevant when the corresponding concepts become relevant during the problem solving.

Decomposing a domain into specialists raises the problem of how they will coordinate during the problem solving process. First, the specialists as a whole are organized, primarily around the "subspecialist-of" relationship. Each task may supply additional relationships that may hold between specialists. Second, each task is associated with a set of strategies which take advantage of these relationships and the problem solving capabilities of the individual specialists. The choice of what strategy to follow is not necessarily a global decision, but can be determined by the specialists in the hierarchy.

2.2. Classification Problem Solving

The classification task is the identification of a case description with a specific node in a pre-determined classification hierarchy. Each node in the hierarchy corresponds to a hypothesis about the state of the situation under consideration. Nodes higher in the hierarchy represent more general hypotheses, while lower nodes are more specific. In medicine, a case description is the manifestations and background information of a patient, and the hierarchy is a classification of diseases and disease classes. For example, the diagnostic expert system MDX [6] attempts to classify a medical case into a hierarchy of cholestatic diseases. Figure 1 illustrates a fragment of MDX's hierarchy. The most general disease, cholestasis in this example, is the head node of hierarchy. More specific cholestatic diseases such as extra-hepatic cholestasis are classified within the hierarchy.

Each disease in the hierarchy is associated with a *specialist* which contains the decision knowledge to evaluate the plausibility of the disease from the case description, specifically the specialist can produce a measurement (called its "confidence value") representing the degree of plausibility of the disease.

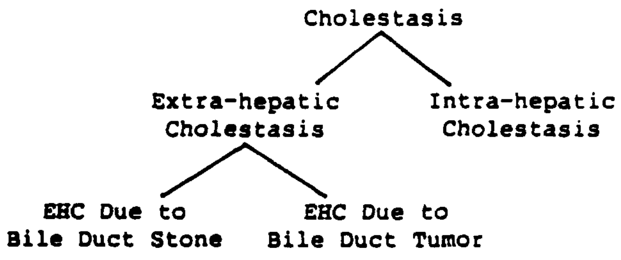

Cholestasis
Extra-hepatic Cholestasis
Intra-hepatic Cholestasis
EHC Due to Bile Duct Stone
EHC Due to Bile Duct Tumor

Figure 1: Fragment of MDX's diagnostic hierarchy

If this value is high enough, the specialist is said to be *established*, i.e., the disease is established as relevant to the case. Each specialist is a problem solver with its own knowledge base.

The basic strategy of classification is a process of hypothesis refinement, which we call *establish-refine*. In this strategy, if a specialist establishes itself, then it *refines* the disease hypothesis by invoking its subspecialists, which also perform the establish-refine strategy. If the confidence value is low, the specialist *rejects* the disease hypothesis, and performs no further actions. Note that when this happens, the whole hierarchy below the specialist is eliminated from consideration. Otherwise the specialist *suspends* itself, and may later refine itself if its superior requests it.

With regard to figure 1, the following scenario might occur. First, the cholestasis specialist is invoked, since it is the top specialist in the hierarchy. Cholestasis is then established, and the two specialists below it are invoked. Extra-hepatic cholestasis is rejected, also eliminating EHC due to stone and bile duct cancer from further consideration. Finally, intra-hepatic cholestasis establishes itself, and invokes its subspecialists.

This simple version of classification does not specify additional specialist-specialist relationships, use recommendation rules, or employ additional control strategies to handle complex situations (e.g. when several nodes are in a suspended state). For discussion on these topics, see Gomez and Chandrasekaran [8] and Sticklen et.al. [13].

2.3. The Role of Classification in Diagnosis

Diagnosis in general is a complex task which requires other kinds of problem solving in addition to classification. One important companion to the classification hierarchy is an intelligent database assistant which organizes the case description, answers queries from the classification specialists, and makes simple inferences from the data [11]. For example, the database should be able to infer exposure to anesthetics from major surgery or exposure to halothane. The classificatory specialists are then relieved from determining how a particular datum could be inferred from another datum.

Another important part of diagnosis is accounting for the patient's manifestations, and producing composite hypotheses when one disease cannot account for all the findings. Josephson et.al. [10] have proposed a method (called hypothesis assembly) to perform these actions in coordination with a classification problem solver. Roughly, the classification problem solver generates plausible hypotheses, and determines the data they can account for. The hypothesis assembler builds and critiques composite hypotheses.

There are several other issues relevant to diagnostic problem solving which we will not consider here, such as test ordering, causal explanation of findings, and therapeutic action. Fully resolving all of these issues and integrating their solutions into the diagnostic framework are problems for future research.

3. CSRL

CSRL is a language for representing the specialists of a classification hierarchy and the knowledge within them. This section briefly describes what CSRL looks like, and the actions that the language can specify. A more detailed description of CSRL can be found in Bylander et.al. [2].

3.1. Specialists

In CSRL, a classificatory expert system is implemented by individually defining its specialists. These definitions include the specialist's relationship to neighboring specialists in the hierarchy (its superspecialist and subspecialists), and the knowledge that the specialist uses during the classification process. Figure 2 is a skeleton of a specialist definition for the Extra-hepatic Cholestasis node in figure 1. The `declare` section specifies its relationships to other specialists. The other sections are examined below.

```
(Specialist Extra-Hep
   (declare (superspecialist Cholestasis)
            (subspecialists Stone BDTumor)
                 ...)
   (kgs ...)
   (messages ...))
```

Figure 2: *Skeleton specialist for Extra-Hepatic Cholestasis*

3.2. Knowledge Groups

The `kgs` section contains a list of knowledge groups, which are used to determine the confidence value of a specialist from the case description. A knowledge group (kg) can be thought of as a group of rules which map a list of values (based on queries to the data base, boolean combinations of queries, other kg's) into a value on a small discrete scale. The knowledge in a specialist can be factored into several kg's, whose values are input to other kg's.

Figure 3 illustrates a table kg named `physical` (other types of kg's in CSRL are comparable to tables and will not be of concern here). The conditions following `match` query the data base, which is independent of CSRL, for whether the patient has cholangitis, colicky pain in the liver, or has been vomiting. Each rule following the `with` is evaluated until one matches. The value corresponding to this rule becomes the value of the kg. For example, the first rule matches if the first and second conditions are true (the "?" means doesn't matter). If so, then 3 becomes the value of the knowledge group. Otherwise, successive rules are evaluated. The value of the `physical` kg can be a condition in another table, whose `match` section might contain tests like `(GE 2)` or `(LT 0)` with the obvious meaning.

```
(physical table
   (match (Present? Cholangitis)
          (Pain? Abdomen Colicky)
          (Present? Vomiting)
     with
          (if     T T ? then 3
           elseif ? T T then 2
           elseif ? T ? then 1
           elseif T ? ? then 1
           elseif ? ? ? then -1)))
```

Figure 3: Example of a knowledge group

3.3. Message Procedures of a Specialist

The `messages` section of a specialist defines a list of message procedures, which specify how the specialist will respond to different messages from its superspecialist. Figure 4 is a message procedure which indicates what to do when a `Establish-Refine` message is received. The `SetConfidence` statement sets the confidence value of the specialist containing this procedure to the value of the `summary` kg. In CSRL, confidence values are taken from a seven-point scale. For convenience we use the integers from -3 to +3. If the specialist establishes itself (`+?` is a predicate which is true if the confidence value of its argument is 2 or 3), then the `for` statement is executed. This statement invokes each subspecialist with an `Establish-Refine` message. "`Self`" and "`subspecialists`" are keywords which evaluate to the name of the specialist and its subspecialists, respectively.

```
(Establish-Refine
   (SetConfidence self summary)
   (if (+? self)
      then (for sub in subspecialists
              do (Call sub with
                    Establish-Refine))))
```

Figure 4: Example of a message procedure

3.4. Levels of Abstraction

These constructs can be used to implement a multilayer evaluation of a disease. At the lowest levels, rules test the values of database queries, and are grouped into kg's. Following this, there can be any number of levels in which several kg's are summarized by another kg. At the highest level, CSRL statements can be used to test the values of kg's and to set the confidence value of the specialist. These levels of abstraction allow a large number of findings to be combined by factoring them into meaningful chunks, evaluating each chunk, and then summarizing the results. This method of combining evidence is borrowed from MDX [7].

3.5. Status

CSRL is implemented on a LISP machine using the INTERLISP-D language [9] and the LOOPS object-oriented programming tool [1]. Each specialist is implemented as a LOOPS class, which is instantiated for each case that is run. The LOOPS class hierarchy is used to specify default message procedures and shared knowledge groups, making it easy to encode a default establish-refine strategy, and letting the user incrementally modify this strategy and add strategies as desired. A graphical interface displays the specialist hierarchy, and through the use of a mouse, allows the user to easily access and modify any part of the hierarchy.

4. Forming a Classification Hierarchy

This section discusses criteria for building a classification hierarchy in the medical domain. An initial difficulty is that a CSRL hierarchy is required to be a tree structure, i.e., a specialist can only have one superspecialist. For medicine this appears to be overly restrictive, since it prevents the implementation of alternative classifications of diseases, e.g., viral hepatitis is an infection, as well as a liver disease. However, there are some advantages for making this restriction in this language.

This restriction simplifies the implementation of the language, as well as the implementation of expert systems in the language. In a "tangled" hierarchy, additional strategies would be required, e.g., when viral hepatitis appears to be relevant during the problem solving, all the superspecialists of the viral hepatitis specialist need to be considered, and if none of them are rejected, then the viral hepatitis specialist needs to take the results of its superspecialists into account. This would increase the complexity of the language, and make it more difficult to use.

It should be pointed out that we are not against tangled hierarchies, per se, but we are against increasing complexity and "knowledge" without achieving a corresponding gain in problem solving ability. CSRL hierarchies are not intended to encode all the "facts" of a domain, but to encode an efficient problem solving structure. There is a need to carefully choose those facts which facilitate classification rather than hinder it. By restricting the structure to a tree, the user is required to face these unpleasant, but necessary decisions. In practice we have found that tree structures are sufficiently powerful for many classification problems. Also, our group is exploring methods which control tangled hierarchies effectively.

4.1. Choosing Between Different Hierarchies

To illustrate the criteria for choosing a CSRL hierarchy, we will present two hierarchies which include viral hepatitis and then evaluate them. The hierarchy in figure 5 differentiates all illnesses into infection, cancer, trauma, etc. Infection is further subclassified into viral, bacterial, and fungal infections. Viral infection has viral hepatitis, viral meningitis, viral encephalitis, etc. as subspecialists. In general, the names of the nodes are abbreviations of more complex statements. The "viral" node in the figure, for example, stands for "viral type of infection." "Viral hepatitis" stands for "hepatitis due to viral infection."

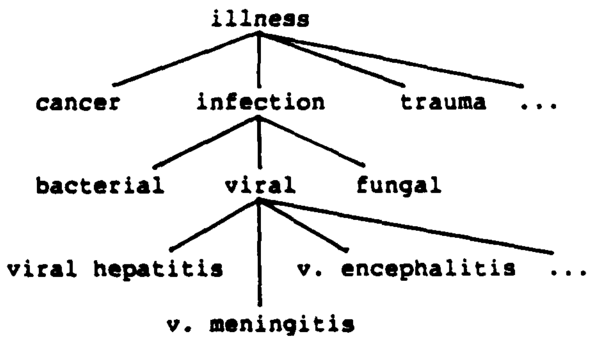


Figure 5: A diagnostic hierarchy containing viral hepatitis

The second hierarchy (figure 6) divides illness into liver, kidney, brain, etc. (location of disease). Liver is subdivided into hepatitis, cirrhosis, cholestasis, etc. (condition due to liver disease). Hepatitis is subclassified into viral and toxic hepatitis (cause of hepatitis).

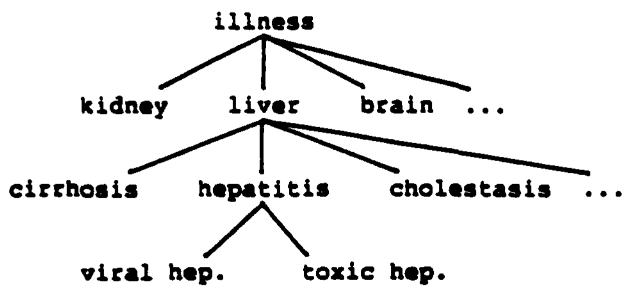


Figure 6: Another hierarchy containing viral hepatitis

We will call the first hierarchy the Infection Hierarchy (IH) and the second the Liver Hierarchy (LH). No claim is made that either one represents the best (or worst) disease hierarchy.

Given the situation where a CSRL knowledge engineer was deciding whether IH or LH was more appropriate, a main consideration is the ability to achieve enough confidence in the presence or absence of a specialist to lead the establish-refine process in the right direction. This property is especially important for higher level specialists. If there is high confidence in such a specialist, it provides focus for further exploration. If a high level specialist can be confidently rejected, a large portion of the hierarchy can be eliminated from consideration. We will call such a specialist an "anchor" specialist.

A specialist is an anchor specialist if there is sufficient evidence to distinguish the specialist from other specialists. Specialists in LH are better than IH in this regard. The presence or absence of liver disease can be evaluated by laboratory tests of the serum and urine, and by clinical symptoms such as jaundice and liver size. Hepatitis can be specifically determined via biopsy. Although there is evidence which is suggestive of infection (such as fever, abnormal white blood count, high percentage of band cells) and viral infection (low white blood count, high percentage of lymphocytes), the evidence does not clearly distinguish infection or viral infection from other diseases. Also, even if these patterns don't exist, infection and viral infection are still reasonable hypotheses. Thus unlike liver and hepatitis, it is difficult to achieve high or low confidence in infection or viral infection.

Evidence for or against a specialist is not very useful if it is not likely to be available to the system when it is running. For example, although a liver biopsy can indicate hepatitis reliably, it is relatively risky and time-consuming to perform, so it is not typically done on patients early in the diagnostic process. Hepatitis then would be a poor anchor specialist if a biopsy was necessary to obtain a reliable decision. Fortunately though, signs, symptoms, generally available laboratory data, and combinations thereof can also reliably indicate the presence or absence of hepatitis.

Since a specialist is not evaluated if any of its superiors have been rejected, the specialist can be implemented assuming that its superiors are plausible. For example, the viral hepatitis specialist in LH could be implemented assuming that the plausibility of hepatitis has already been established. Besides being an efficiency measure (the viral hepatitis specialist doesn't have to redo the work of the hepatitis specialist), this context allows evidence which distinguishes viral hepatitis from toxic hepatitis to be used to increase confidence in viral hepatitis. For example, if we know that hepatitis is plausible *and* that the patient has not been exposed to any hepatotoxins, then viral hepatitis will be implicated. One desirable quality of a CSRL hierarchy then is that its specialists provide good differential contexts. The specialists in LH are better than IH in this regard.

The existence of a specialist should imply the existence of its superspecialist. Normally this is not a problem (cases of viral hepatitis, for example, are by definition cases of infection), but in some instances this is meaningful to consider. Suppose that we propose jaundice as a subspecialist of liver (to be precise, "jaundice due to liver disease"). Since hepatitis can cause jaundice, it might be appropriate to propose hepatitis as a subspecialist of jaundice. However, problems will occur because there are a significant number of hepatitis cases in which jaundice never appears, or appears late in the process. This hierarchy will misdiagnose these cases simply because the hierarchy is improperly formed. To remedy this problem, the hepatitis specialist should be renamed "hepatitis causing jaundice," and there should be other specialists elsewhere under liver which correspond to other "kinds" of hepatitis.*

Finally, a hierarchy is usually designed to make certain diagnostic statements. If you intend to develop an expert system which can directly conclude infection, then IH better satisfies that purpose. If you need a system which has hepatitis and cirrhosis as specialists, then LH is better.

From these criteria, the kinds of questions that one should ask about a CSRL hierarchy are:

*This type of redundancy is inevitable in any CSRL hierarchy. IH, for example, must contain liver diseases across separate branches of the hierarchy, while LH must contain infections in separate branches.

- Are the specialists anchor specialists?

- Is the evidence that is needed for a confident decision normally available to the system in the early stages of its problem solving?

- Do the specialists provide good differential contexts?

- Does the presence of each specialist logically imply the presence of its superspecialist?

- Does the hierarchy make the diagnostic statements that it is intended to?

4.2. How a Specialist Decomposes

The basic relationship between a specialist and any of its subspecialists is that the subspecialist represents a more specific hypothesis about the diagnostic state of the patient. The hierarchies in figures 5 and 6 exemplify this relationship. A simple method to subclassify or *decompose* a specialist into its subspecialists is to ask the domain expert what should be considered next after that specialist is established. For example, a physician who is asked what should be done after establishing hepatitis might say to then differentiate between viral and toxic hepatitis.

A more careful approach would also look at what kinds of additional information can be used to decompose a specialist. By considering these, one can determine what decompositions are possible and evaluate them using the criteria discussed above. Another motivation is to more precisely define what a specialist can be, i.e., it represents a diagnostic statement that specifies one or more of the following attributes.

- A specialist can indicate the location of a disease. For example, in figure 6, illness is decomposed into the location of illness — liver, kidney, brain, etc.

- The system that the disease affects can be specified. Illness, for example, could be alternatively decomposed into circulatory system, urinary system, nervous system, and so on.

- A specialist can specify the pathologic condition or syndrome associated with the disease. For instance, liver in figure 6 is decomposed into various conditions of liver disease — hepatitis, cirrhosis, cholestasis, etc.

- The underlying process of the disease can be indicated. Liver could be decomposed instead into liver infection, liver cancer, toxic liver, etc.

- A specialist can state the type of cause that is involved with the disease. This is the parameter used to decompose infection into viral, bacterial, and fungal infection.

- A specialist can also point out the specific entity which is involved. Viral hepatitis could be decomposed according to specific viruses, e.g., hepatitis A and B, Epstein-Barr, mumps, etc.

The last three attributes could be grouped as etiological factors.

This is not intended to be a complete list of classification attributes. Other possibilities include temporal factors (e.g., acute vs. chronic) and physical distribution (e.g., focal vs. diffuse). Also we have not precisely indicated the causal relationships are possible among these attributes, e.g., inflammation (a condition) could be viewed as either causing disease, or being caused by disease.

As a result of our experience, we suggest that a diagnostic hierarchy should be formed according to the following heuristics.

- The top levels of the hierarchy should specify the locations or systems containing the diseases.

- The middle levels should specify the conditions caused by the diseases, the syndromes associated with the diseases, and/or the underlying processes causing the diseases.

- The bottom levels should identify the specific etiological agents of the diseases, such as the microorganism causing the infection.

5. Establishing a Specialist

The most important part of implementing a specialist in CSRL is the knowledge base that determines the specialist's plausibility. CSRL uses a 7-point scale of -3 to +3 to indicate the relative confidence in the specialist. -3 means that the specialist is not plausible. +3 means that the specialist is highly plausible. 0 means that the evidence is neutral with respect to plausibility. The rest of this section discusses the criteria for selecting the evidence that a specialist uses, and for using knowledge groups to determine the specialist's confidence value. We will use a specific example, in this case cholestasis, to help clarify the discussion.

5.1. Selecting Evidence

We have already mentioned in passing the kinds of evidence that are desirable to have. The best evidence is specific or sensitive to the specialist. From a finding that is specific to the specialist, the specialist can be definitely established. If a sensitive finding is not present, the specialist can be definitely rejected. These conclusions can also be inferred from specific and sensitive combinations, or *patterns*, of evidence.

Next to consider are patterns of evidence which differentiate the specialist from its siblings in the diagnostic hierarchy. If a pattern is specific in relation to other siblings of the specialist, then the confidence in the specialist will be increased. We gave an example earlier for differentiating viral hepatitis from toxic hepatitis. For cholestasis, lack of liver cell damage can differentiate it from other liver conditions at the same level in our example hierarchy.

Finally, the expected patterns of the specialist should be considered. Confidence in the specialist should increase or decrease depending on whether the expected pattern is present or not. The amount of confidence (either for or against) will depend on the degree that the pattern is sensitive to the specialist and unexpected in other specialists.

Each kind of evidence above is used to differentiate the specialist from other specialists. Evidence which does not satisfy this function should not be used. For example, each kind of cholestasis (e.g., bile duct stone causing cholestasis) will have the expected patterns of cholestasis, but these patterns are useless for differentiating among different causes of cholestasis. See Price and Vlahcevic [12] for additional discussion on the use of evidence in medical reasoning.

5.2. Knowledge Groups

Each important pattern of evidence associated with a specialist should be represented by a knowledge group which indicates how well the findings fit the pattern. We suggest the following process for implementing a set of CSRL knowledge groups. First, the patterns of evidence which will be implemented as knowledge groups should be decided upon. Next, for each knowledge group, the findings that it will evaluate should be selected. A confidence value should then be assigned to each pattern of findings. Like any program, knowledge groups should be tested and debugged on appropriate data, i.e., actual cases. The following sections examine the steps of this process (except for test and debug) for cholestatic disease. We will assume that cholestasis is a subspecialist of liver disease (as in figure 6).

5.3. Determining the Set of Knowledge Groups

The important part of this step is to determine patterns of evidence which correspond to *intermediate hypotheses* related to the diagnosis of the disease. The result from the knowledge group which evaluates this kind of pattern should be the level of confidence in the intermediate hypothesis.

A common pattern for most specialists is whether it can be directly observed (for example by some visualization process). For cholestasis, observation of biliary tree obstruction or bile stasis is possible by evaluating relevant xrays or liver biopsies.

We recommend that the other knowledge groups of a specialist be related to the primitive processes of the disease. For cholestasis, we would examine the major consequences of bile obstruction, and propose knowledge groups for each of the following:

1. Are normal amounts of bile reaching the duodenum? This knowledge group determines if bile is performing its role in the digestive process.

2. Is bile accumulating in the liver and bloodstream? When the bile cannot be excreted via the biliary tree, it accumulates in the liver, and its constituents enter the blood.

3. Is there bile duct damage? The kind of bile duct damage which is associated with obstruction is evaluated.

4. Is there liver cell damage? Lack of liver cell damage can differentiate cholestasis from other subspecialists of liver disease.

In addition, characteristics which predispose the patient to the disease should be evaluated. This is not very relevant in the diagnosis of cholestasis (for specific causes of cholestasis though, this becomes more important). However, for most kinds of infections, e.g., determining whether the patient has been *exposed* to the microorganism or is *susceptible* to the infection are important factors to consider in diagnosis.

Another method for dividing a specialist into knowledge groups is to have each knowledge group evaluate a particular kind or source of evidence. Using this method, cholestasis could be broken up into knowledge groups for physical, laboratory, and xray evidence. These would respectively evaluate the signs and symptoms, laboratory data, and radiographic data. This often corresponds to the sequence of evidence collection that physicians perform in evaluating patients, and is useful for diseases whose processes are not clearly delineated, or when these processes are not known to the physician acting as the domain expert.

5.4. Determining the Findings of a Knowledge Group

Once the knowledge groups have been selected, the findings that each knowledge group will evaluate must be determined. Here, we will concentrate on knowledge group 2 from the list above, which examines the findings when bile accumulates in the liver and blood. Findings which are related to this decision are jaundice, elevated serum bilirubin, conjugated serum bilirubin, bilirubinuria, pruritis, elevated serum lipids, xanthoma, and large, smooth liver.

When the bile is unable to flow into or through the bile ducts, it accumulates in the liver, and enters the bloodstream. Conjugated bilirubin and lipids are major components of bile, so elevated serum bilirubin and serum lipids occur. When serum bilirubin is highly elevated for a period of time, it accumulates in skin tissue, resulting in jaundice. Since conjugated bilirubin is water-soluble, the kidneys filter it out and excrete it, resulting in bilirubinuria. Pruritis is usually attributed to elevated serum bile salts, and their deposition in skin tissue. Xanthoma is a result of the excess lipids which accumulate in the skin over a period of time. A large, smooth liver is a consequence of the bile accumulating within the liver.

The next stage is to combine findings which are closely related into a single condition. This will reduce the number of conditions of the decision table, and simplify the process of filling it in. Jaundice is an indication of elevated serum bilirubin and thus can be combined with elevated serum bilirubin. Similarly, bilirubinuria is an indication of elevated conjugated bilirubin. Also, elevated serum lipids and xanthoma are both results of bile fats accumulating in the blood, so these can be combined. Thus the conditions of this knowledge group will be:

- Is jaundice or elevated serum bilirubin present?
- Is bilirubinuria present or is the conjugated percentage of serum bilirubin high?
- Is xanthoma or elevated serum lipids present?
- Is pruritis present?
- Is the liver large and smooth?

5.5. Assigning Confidence Values

The last step we will discuss is assigning confidence values for each combination of the values of the above conditions. We will assume that the possible values of each condition is true, false, or unknown (T, F, or U). It is important that the meaning of the knowledge group's result be clear. In this knowledge group, we want the result to indicate the level of confidence in the hypothesis "bile is accumulating in the liver and blood." How this value affects confidence in cholestasis is determined by another knowledge group, and will not be discussed here.

Essentially, this is a process of asking the domain expert for the level of confidence for each combination, and insuring that the overall knowledge group is consistent. This initially appears intimidating since there are $3^5=243$ possible patterns for this knowledge group. However, the number of rows in a decision table will be much smaller since the values of some conditions will not matter for some patterns. Also, this problem can be easily handled by dividing the patterns into separate cases according to the value of a particular condition. For example, figure 7 could be the segment of the table where the first condition (elevated bilirubin) is false. Note that the meaning of the last row, which just checks to see if the first condition is false, depends on what conditions have been taken care of in the previous rows. Also, this figure assumes the 7-point confidence scale described above. Finally, note that except for coding the conditions properly, which depends on how the data base is implemented, it is trivial to translate the table into CSRL.

elev. bili.	conj. bili.	elev. fats	prur.	large smooth liver	confidence value
F	T	?	T	T	-1
F	F	?	?	?	-3
F	?	F	?	?	-3
F	?	?	F	?	-3
F	?	?	?	F	-3
F	?	?	?	?	-2

Figure 7: Fragment of table determining bile accumulation

6. Conclusion

Designing a diagnostic system in CSRL requires emphasis on organizing knowledge to reflect the structure of diagnostic reasoning. Knowledge is represented at various levels of abstraction -- from the classification hierarchy to rules within knowledge groups. This is in contrast to MYCIN and similar rule-based systems which are constructed as collections of rules without any higher organizing constructs. A classification hierarchy is designed so that establish-refine problem solving performs well on it. The knowledge base of a specialist is organized according to how groups of evidence support or refute intermediate hypotheses which affect confidence in the specialist.

Acknowledgments

We thank B. Chandrasekaran for his direction and support of the CSRL project. Sanjay Mittal originally conceived of CSRL and was one of its principal designers. We also thank the rest of the AI group at Ohio State for their comments and their encouragement. The development of CSRL is funded by a grant from the Battelle Memorial Laboratories University Distribution Program, and experimentation and application in different domains is supported by AFOSR grant 82-0255 and NSF grant MCS-8305032. Jack W. Smith is supported by Career Development Award K04 LM00003 from the National Library of Medicine.

References

1. D. Bobrow and M. Stefik. The LOOPS Manual. KB-VLSI-81-13, Xerox Palo Alto Research Center, 1982.

2. T. Bylander, S. Mittal, and B. Chandrasekaran. CSRL: A Language for Expert Systems for Diagnosis. Proc. Eighth International Joint Conference on Artificial Intelligence, IJCAI, Karlsruhe, 1983, pp. 218-221. An expanded version will appear in the special issue of Int'l Jrnl. of Computers and Mathematics on practical artificial intelligence systems..

3. B. Chandrasekaran. Towards a Taxonomy of Problem Solving Types. *AI Magazine 4*, 1 (Winter/Spring 1983), 9-17.

4. B. Chandrasekaran. Expert Systems: Matching Techniques to Tasks. In W. Reitman, Ed., *Artificial Intelligence Applications for Business*, Ablex, Norwood, New Jersey, 1984, pp. 116-132.

5. B. Chandrasekaran. Generic Tasks in Expert System Design and Their Role in Explanation of Problem Solving. Proc. National Academy of Sciences/Office of Naval Research Workshop on AI and Distributed Problem Solving, 1985. To appear.

6. B. Chandrasekaran and S. Mittal. Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems. In *Advances in Computers*, Academic Press, New York, 1983, pp. 217-293.

7. B. Chandrasekaran, S. Mittal, and J. W. Smith. Reasoning with Uncertain Knowledge: The MDX Approach. Proc. Congress of American Medical Infomatics Association, AMIA, San Francisco, 1982.

8. F. Gomez and B. Chandrasekaran. Knowledge Organization and Distribution for Medical Diagnosis. *IEEE Trans. Systems, Man and Cybernetics SMC-11*, 1 (January 1981), 34-42.

9. *Interlisp Reference Manual*. Xerox, 1983.

10. J. Josephson, B. Chandrasekaran, and J. W. Smith. Assembling the Best Explanation. Proc. IEEE Workshop on Principles of Knowledge-Based Systems, IEEE Computer Society, Denver, 1984, pp. 185-190.

11. S. Mittal, B. Chandrasekaran, and J. Sticklen. Patrec: A Knowledge-Directed Database for a Diagnostic Expert System. *Computer 17*, 9 (1984), 51-58.

12. R. Price and Z. Vlahcevic. Logical Principles in Differential Diagnosis. *Annals of Internal Medicine 75*, 1 (July 1971), 89-95.

13. J. Sticklen, B. Chandrasekaran, and J. Josephson. Control Issues in Classificatory Diagnosis. Proc. Ninth Int'l Joint Conf. on Artificial Intelligence, IJCAI, Los Angeles, 1985. To appear.

Appendix H

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

Technical Report

April, 1985

PLAN SELECTION IN DESIGN PROBLEM-SOLVING

David C. Brown
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

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

Presentation at The Society for AI and the Simulation of Behavior (AISB) 1985 Conference, Warwick, England, April 1985. Appears in the Proceedings of The Conference.

PLAN SELECTION IN DESIGN PROBLEM-SOLVING

David C. Brown

Computer Science Department,
Worcester Polytechnic Institute,
Worcester, MA 01609, USA

B. Chandrasekaran

AI Group, CIS Department,
Ohio State University,
Columbus, OH 43210, USA

Abstract

The AIR-CYL expert system for mechanical design is based on a hierarchy of active design specialists, each of which solves part of the design problem and cooperates with the other specialists. Every specialist uses a small collection of preformed plans that represent ways to achieve the subproblem for which that specialist is responsible. When a specialist is asked to design, it selects a plan from its collection in a situation dependent manner. We discuss this plan selection process, and the types of knowledge used.

Keywords:
Expert Systems, Plans, Design, Plan Selection, Problem-solving.

1 INTRODUCTION

This research is concerned with the design of mechanical components, and views design as a problem-solving activity. We will present a theory of design that explains the activity of a human designer when solving a problem that falls into a particular subclass of mechanical design. This paper concentrates on one aspect of design activity, that of selecting preformed design plans.

Design activity in general has many components; such as planning, the use of prestored plans, refinement of descriptions, and the use of large amounts of knowledge. Not all designing involves all of these. We have identified three classes of design activity which vary according to their problem-solving components [CHAN83]. Our work refers only to the third class, where the designer knows in advance exactly what knowledge and what type of problem-solving will be required during the design. An important piece of knowledge is that at every stage of the design the designer knows what sequences of design steps are appropriate. Class

3 design occurs when a specific object has been designed many times before, each time with different requirements. Other classes of design are more general and involve other types of problem-solving, such as planning.

In order to manage the complexity the design is broken into roughly independent subproblems [SIMO69]. The hierarchy reflects the way that the designer thinks about the object during design, and, consequently, it shapes the design process. Our theory hypothesizes that design activity is organized around this conceptual hierarchy, where each concept is active in the design, and may be considered to be a specialist about some portion of the design. Each specialist does the same kind of design problem-solving but uses different knowledge.

For every subproblem in a class 3 design the designer has a small set of Plans that can be followed to produce a piece of the design. Consequently, each specialist has its own set of plans from which to select depending on the current stage of the design. The plans may request portions of the design from other specialists lower in the hierarchy. Thus the specialists solve the problem cooperatively. Tasks, which are pieces of design knowledge local to a specialist, can be used in plans to make small additions to the design. Tasks use Steps to decide the value of each attribute for which it is responsible. For example, a hole might be designed by a Task, while a Step would decide the radius. Constraints are placed at various places throughout the design knowledge to check the progress of the design and ensure its validity. A Design Data-Base contains the current state of the design.

The complete design process proceeds by first obtaining and checking requirements for consistency. It then does rough-design to establish whether full design is worth pursuing. If the rough-design succeeds, then the full design is attempted by requesting a design from the top-most specialist. Communication between active design agents is done by passing messages that give instructions and report on success or failure.

If a design agent fails, a redesign phase is entered until the problem can be fixed and design can continue. Different types of agents have different failure recovery strategies. These result in backing-up over prior design decisions in a manner which is dependency-based.

To demonstrate and test our theory of design, we have developed a working expert system, AIR-CYL, that does Air-cylinder design [BROW83, BROW84A, BROW84B]. The system closely mirrors the activity of a human designer solving the same problem. The system will design a particular type of Air-cylinder according to some set of user given requirements. The system takes about 5 minutes to design an

Air-cylinder given about 20 requirements using a DECsystem-20.

To facilitate the building of the AIR-CYL system, and class 3 design problem-solvers in general, a language called DSPL (i.e., Design Specialists and Plans Language) has been designed and implemented. DSPL has been used to capture the Air-cylinder design knowledge. Close attention has been paid to the many different types of knowledge that are used during a design of this kind. DSPL allows different types of design knowledge to be described separately. This is in contrast to design systems based on more uniform representations [MCDE82, STAL76, MOST83, KOWA83].

2 SPECIALIST ACTION

A specialist is responsible for supervising some portion of the design. Specialists consider design situations and produce courses of action which lead to changes in the state of the design. Specialists are responsible for the flow of control during design problem-solving. The selection and execution of plans provide these control decisions.

In contrast, tasks supervise the design activity carried out by steps. A task makes no control decisions that affect the overall problem-solving flow and always attempts to carry out the same series of actions. The main role of the task in the system is an organizational one -- that of grouping some related steps and executing them in order.

2.1 Plan Action

A plan is the result of past planning by a human designer. It is the result of prior decisions about the flow of control in a portion of the design for a given situation. A plan could, for an auto example, specify that the engine should be designed first, followed by the suspension and then the body, with each possibly being preceded by a rough design.

A plan is executed by testing its applicability conditions and then executing each plan item in turn. A plan item can be a task, the testing of some constraint, a design or rough-design request of a specialist, or an indication that some specialists should be used in parallel.

By selecting a plan the specialist is refining the plan that calls it. The specialists in the hierarchy act together as if they were gradually "inserting" plans into other plans in order to "construct" the plan that will produce a successful design. Figure 1 demonstrates this and shows the plan that has been selected by specialist S0 being refined by specialists S1, and S2, while S3 refines the plan in S2. The Ti are tasks.

```
      <  S1    ;    T1    ;    S2  >
        /  \                  /  \
      /      \              /      \
    /          \          /          \
   <  T2  ;  T3 >        <  S3  ;  T4 >
                           /  \
                         /      \
                       /          \
                      <  T5  ;  T6 >
```

Figure 1: Plan Refinement

3 PLAN SELECTION

Plan selection depends on three types of information: the qualities of the plans themselves, the requirements from the user and the design and its history). We propose a general method of plan of selection that responds to all factors.

A designer may have very simple selection criteria, such as "if there is a plan that hasn't been tried yet then try it", or they may be very complex. For example, "if there are some plans that look perfect for the situation then if plan X is amongst them then use it, otherwise pick the one that has been the most reliable in the past, unless it contains the item that failed in the last plan". In general, complex analysis of prior failures may precede a plan selection, but at present we do not have enough understanding of these processes to be certain that we have provided the language necessary to express them.

The selection process will select one plan from several. Some plans will not be suitable for consideration. Others will, but with various degrees of suitability. Selection then will occur after the individual plans available in the specialist have been evaluated for suitability. Consequently, we will divide the whole selection process into evaluation and selection processes, each with its own knowledge. It may be that the processes of evaluation and selection are intermixed. For this to occur there would have to be knowledge in the form of "suggestions" about which evaluations to do before others. We have currently made the simplifying assumption that all plan evaluations are made prior to selection.

First we will discuss the process of selection and then the knowledge that can be used during plan selection. An example of the system doing plan selection can be found in Appendix A and the reader is urged to follow that example to obtain a better understanding of the problem-solving involved. A more detailed explanation of DSPL can be found in [BROW85].

3.1 The Selection Process

```
                 Selector
                 /   |   \
                /    |    \
               /     |     \
              /      |      \
             /       |       \
      Sponsor     Sponsor     Sponsor
         |           |           |
       PLAN1       PLAN2       PLAN3
```

Figure 2: Plan Selection

Plan selection divides into two parts -- first the recommendation of those plans that are candidates for use, and second the selection of a plan from the set of candidates. Each plan has associated with it a Sponsor and some information about the qualities of the plan. It is the job of the plan's sponsor to use its current situation, the qualities of the plan, the user's preferences about qualities, and special case information to make an evaluation of whether its plan is a suitable candidate for selection. It will give an estimation of the plan's suitability. The Selector has the job of collecting the responses from the sponsors, promoting or relegating if necessary, and selecting one plan for use.

Sponsors will respond with some scale of suitabilities such as (Perfect, Suitable, Don't-know, Not-suitable, Rule-out). Those ruled out will never get used, unless the knowledge encoded in the Selector gives strong reasons to do so. If more than one plan is Perfect then the selector will pick one. If none are Perfect, then a Suitable plan might be used. The Selector can, depending on its stored knowledge, do plan suitability relaxation in order to select from amongst the lower rated plans.

In many situations, there will be few plans, or the suitabilities will always be the same, and, consequently, this kind of effort during plan selection is not always warranted. In these cases the order can be fixed. In the AIR-CYL implementation, plans are selected in the order the designer prefers.

3.1.1 Sponsors - Figure 3 shows the DSPL for an example of a Sponsor.

```
(SPONSOR
 (NAME    Sponsor1)
 (USED-BY ExampleSelector)
 (PLAN    Plan1)
 (COMMENT "Sponsor returns suitability")
 (BODY
  COMMENT "rule out plan if already tried"
  REPLY (IF (ALREADY-TRIED? PLAN) THEN RULE-OUT)
  COMMENT "rule out plan if plan Plan3 failed"
  REPLY (IF (ALREADY-TRIED? 'Plan3) THEN RULE-OUT)
  COMMENT "use qualities to get suitability"
  Qualities
   (TABLE (DEPENDING-ON
           (RELIABILITY-REQS)(COST-REQS))
       (MATCH
        (IF (Reliable  Cheap)THEN PERFECT)
        (IF (Medium      ?  )THEN SUITABLE))
         (OTHERWISE RULE-OUT))
  COMMENT "was Task2 the last failure?"
  Agent   (EQUAL 'Task2 LAST-FAILING-ITEM)
  COMMENT "now use vbls to get suitability"
  REPLY
   (TABLE (DEPENDING-ON
           Agent Qualities)
       (MATCH
        (IF ( T      ?      )THEN RULE-OUT)
        (IF ( ?  PERFECT  )THEN PERFECT)
        (IF ( ?  SUITABLE )THEN SUITABLE))
         (OTHERWISE DONT-KNOW))
))
```

Figure 3: Sponsor "Sponsor1"

Each sponsor will be associated with only one plan as it will have knowledge about the applicability of that plan to different situations. Every plan has a sponsor. The output of a sponsor is a classification, i.e., one of the categories from the scale of suitabilities. The inputs to a sponsor are the various sources of knowledge already outlined above. Notice though that some knowledge, for example plan complexity, will belong only in the selector, as it will be used to choose between equally suitable plans.

In a sponsor, evidence needs to be accumulated about the plan's suitability for use. Some information can be used to determine the suitability of a plan quite quickly, while other pieces of information need to be put together to build a picture of suitability. Consequently, two kinds of expressions of knowledge are needed: one with immediate result and one that accumulates a result.

The immediate form of knowledge is a rule -- for example "if this plan has already been executed during this use of the specialist then it must have failed and should not be considered at this point"; that is, its suitability is "Rule-out". The accumulated form of knowledge will weigh the answers to several "questions", such as "is a COST=Cheap required?", and combine them to produce a suitability value according to some designer-dependent logic. This is not meant to imply that predicate calculus is used, but rather that under some circumstances the designer will make "suitable" AND "perfect" produce "perfect", while in others it will produce "suitable".

As there are several different types of knowledge we will propose that evidence is accumulated for each type, e.g., qualities, and then these pieces of evidence are combined to form an overall suitability to represent the plan. This approach has already been demonstrated in the CSRL language [BYLA83] where diagnostic knowledge is factored into knowledge groups.

3.1.2 The Selector - Figure 4 shows the DSPL for an example of a Selector.

```
(SELECTOR
 (NAME    ExampleSelector)
 (USED-BY Example)
 (TYPE    Design)
 (USES    Sponsor1 Sponsor2 Sponsor3)
 (COMMENT "Selector returns name of plan")
 (BODY
   COMMENT "if Plan2 is perfect use it"
   REPLY (IF (MEMBER 'Plan2 PERFECT-PLANS)
         THEN 'Plan2 )
   COMMENT "if there are perfect plans
              use them in preferred order"
   REPLY (IF PERFECT-PLANS
           THEN (DESIGNER-PREFERENCE
                    PERFECT-PLANS)
           ELSE NO-PLANS-APPLICABLE)
))
```

Figure 4: Selector "ExampleSelector"

Each selector will be associated with a specialist and will have a collection of subordinate sponsors. A selector contains specialist dependent knowledge. The input to a selector is the collected outputs from all of the sponsors that report to it, and the state and history of the design. A sponsor's output consists of the name of its plan and a suitability. So, for example, the information received might be:

{(plan1, perfect) (plan2, suitable) }.

The output from the selector consists of either the name of the plan selected for execution by the specialist, a failure due to there being no plans appropriate, or a failure due to all plans having been tried already.

The exact operation of a selector will depend on the personal preferences and experiences of the designer whose knowledge we are trying to capture. The "normal" knowledge would take the plans ranked as "perfect" and choose one. If there are no perfect plans then "suitable" ones will be considered, and so on. Exactly how many of the suitability categories will be considered as acceptable can be made to depend on any appropriate factor: for example, on the position of the specialist in the design hierarchy. A specialist at the lowest extremes can afford to try plans that are less appropriate, as not much effort is being wasted if they fail (i.e., there aren't many agents below). However, at higher levels there are many specialists below and any relaxation of standards could be very costly.

If several plans appear to equally suitable the designer is most likely to pick the one that has performed the best in the past. That is, the designer has an order of preference. Another approach is to compare some quality (for example COST) and pick those with the best value (COST=Cheap). This can be repeated with other qualities (such as WEIGHT=Light) until one plan remains. Notice that there is no need to prescribe a global ordering for qualities using this method, as orderings will be local to specialists and sensitive to the situation.

Knowledge that can be used during selection includes plan complexity, existing preference, position in the hierarchy, special rules about the use of particular plans (e.g., "if plan A is perfect and it hasn't been used before then use it before any others"), dependencies between attributes of the design, and knowledge about past plan failures.

3.1.3 Additional Strategies - One option available to a designer during plan selection is Plan Reordering. If there is evidence to suggest that the first part of the plan will succeed, but some subsequent item may fail, then less effort would be wasted if that dubious plan item could be executed earlier. Another technique is to not immediately discard the partial design achieved by the successful part of the last failing plan. By reasoning about the dependency relations carefully it is possible to selectively discard or keep parts of the design done by the previous plan so that that work need not be repeated.

3.1.4 Plan Selection In AIR-CYL - It may be that plan selection really isn't that much of a problem as it immediately appears. Due to the decomposition into specialist. If there are few plans to choose from then selection mnot be very cof there are few plans to choose from then selection may not be very complex.

Suppose that there are two plans in every specialist in a fully specified AIR-CYL system. Given the current specialist hierarchy, there would be about 130 different sequences of plans that can be followed in order to achieve the design. A very large number of actual designs can arise depending on the actual values chosen. So even with a small hierarchy and a small number of plans at each point there are still a very large number of designs captured. The human designer will *gradually form* preferences and may actually do very little work during plan selection.

3.2 Qualities Of Plans

It is not possible to prescribe in advance exactly which pieces of knowledge will be used in which cases. We are arguing that these types of knowledge exist, that an Expert System builder should be provided with language in which to express them, and that this knowledge must all be available to be used by the plan selection mechanism. The theory acknowledges that there are these types of knowledge but does not and cannot describe exactly how each will be used, as it will vary depending on the domain and specialist involved.

Plans can have qualities associated with them. These qualities can refer to some attribute of the plan, some attribute of the design, or some attribute of the object being designed. A similar set of qualities were used by Friedland [FRIE79] in his version of the MOLGEN system.

* Precision: A plan with precise measurements is more expensive in manufacturing terms, and may be harder to design as there is less "slack" in the plan.

* Convenience: Convenient plans will have easier calculations, less difficulty with tolerances, fewer elements, fewer other specialists used, and less anticipated trouble.

* Reliability of design: Some plans will be associated with reliable products as they capture methods that produce reliability.

* Reliability of plan: A designer will know the likelihood of success for a plan. This will have a general component, (i.e., works fairly often), and a context dependent component, (i.e., fails often if Aluminium is the material). A plan that works often will be considered reliable.

* Cost: An expensive plan produces an expensive product.

* Designer's time: If the plan takes a long time to follow due to many calculations, many steps, many questions of the user, or many catalogue lookups it will be noted as taking a lot of the designer's time.

* Manufacturer's time: A note is made if a plan takes a lot of the manufacturer's time.

* Plan Complexity measures:

  - Length of Plan: If all else is equal then the designer can be expected to choose the shortest plan.

  - Complexity from Structure: Assuming that it is preferable to select a plan that is in some way "simpler" than another, a "cheap" estimate of complexity is useful. It is possible to obtain some crude measure of the complexity of a plan by using just its surface syntax -- i.e., without detailed knowledge of the structure or action of the components of that plan.

  - Complexity from Dependencies: Another measure takes into account the designer's knowledge of agent-agent dependencies. Suppose specialist S1 has a dependency measure of 5 (i.e., five attributes depend on it), S2 has a measure of 3, task T1 has 2 and T2 has 1. Consider two plans, < S1 ; T1 > and < S2 ; T2 >. The first plan, using a simple sum, has a measure of 7, while the second has a measure of 4. Thus the first plan has more ramifications if adopted, and might therefore be worth avoiding.

This, and surface complexity, are presented in more detail below.

* Manufacturability: If the processes involved require much skill, unusual machines, special techniques, special materials, and unusual attention to detail then the plan would be classified as difficult to manufacture.

* Weight: If the manufactured product falls toward the high end of the range of reasonable weights the plan would be classified as heavy.

3.3 Situation Factors

In addition to plan qualities, selection depends on the situation at the time of selection. Information that may be relevant includes :-

1. Plan selection information from above or below.

2. Which plans were selected already by this specialist and how they failed.

3. The current state of the design (i.e., values chosen).

A specialist can select a plan according to some preferred quality. In this situation it is good to have specialists below select plans that are in some way compatible -- there is little point selecting a COST=expensive plan inside a COST=cheap plan. A specialist in a plan may be passed information on activation to allow this strategy. There may also be more subtle plan interactions, where from experience it has been discovered that while in plan x selection of plan y is to be preferred.

The history of the selection process is important for subsequent selections. One does not wish to select the same plan again, or one that is similar to others that have failed. The structure and properties of plans that failed should be abstracted out and used to indicate which others to avoid.

The way plans failed is also of use. Not only can there be knowledge such as "If plan A failed then so will plan B", but also more subtle knowledge such as "If plan A failed due to task 1 then plan B will fail", or even "If plan A failed due to xyz being too large then plan B will fail". Knowledge can also be in a positive form so that failure of a plan suggests the selection of another. All of this needs a representation of reasons for a plan's failure to be associated with the plan. The plan item that caused the failure and an abstracted form of its reasons for failure should be available.

Plan selection may also depend on the past choice of values. For example, "If xyz < 0.5 then use plan B". As small variations in the component can be introduced by different tasks, selection can depend on this too -- "If cross section of connecting rod is rectangular then try plan A otherwise try plan B". These kinds of selection rules could actually be expressed as constraints in the plan, so that the use of a plan would be accepted or rejected on entry. However, after continued use this knowledge would migrate so that it could be used during the selection process.

3.4 Plan Complexity

The complexity of a plan can be known and associated with the plan. It can be accessed and used during plan selection.

3.4.1 Plan Complexity From Structure - The components of a plan are tasks, constraints, or specialists. If we take the constraint as the major cause of problems in the system, then we can assign a rough complexity to each type of agent depending on how many constraints we expect to find in each on average. Assume that on average failures will occur in the middle of a plan. This means that items towards the beginning of the plan will on average be more often involved in failure handling and redesign. If we use position-in-plan as weights then some rough measure of the influence of position in the plan can be included. This measure is a surface indication of complexity taking into account design and redesign. This method also builds in the length of the plan as a factor. Longer plans are more complex.

3.4.2 Dependency As A Measure Of Plan Complexity - A plan has a dependency measure. That is, some estimate of the number of attributes that depend on the attributes for which this plan decides values. Thus a plan might be worth avoiding if it has more affect on the design. Just as with the surface structure measure of complexity we want to include the possibility of failure and its implications. If we assume that on average failure occurs in the middle of the plan then we know that agents towards the beginning of the plan will tend to be involved in redesign more often. A redesign will affect more of the design if the agents used in redesign have a high dependency measure. As before, we can calculate a weighted dependency.

It must be stressed that it is not being suggested that numerical measures such as these actually exist, but rather that, with experience, some feeling for the complexity of a particular plan can be formed based merely on general knowledge about plans, and that these complexities can be used and compared. Plan selection can use symbolic complexity measures as one of the available sources of knowledge. A designer will probably have "worked out" some "value" for the complexity over a period of time and will use that without regard to how it was originally obtained.

4 SUMMARY

This paper has presented an analysis of plan selection in design problem-solving. Plan selection is a knowledge-based process involving two different types of knowledge. A Sponsor uses its knowledge to estimate a suitability for the plan it represents given the current situation. A Selector uses the information from its sponsors and its knowledge to select the plan to be tried next. Collections of plans, their sponsors and their selector are associated with a specialist. Selection takes place in the context of a particular subproblem in the design. This theory has been included in the DSPL language which is currently under development. A version of DSPL has been used to implement a problem-solver to design a small air-cylinder. More research is needed to identify exactly what kinds of knowledge are consistently used by designers to select their plans.

Acknowledgments:
This work was supported at Ohio State University by AFOSR grant #82-0255. We would also like to acknowledge the cooperation of the AccuRay Corporation and Pete Schmitz, and the assistance of Dave Herman.

5 APPENDIX A

This is an edited form of a trace generated by an example written in DSPL. This trace shows a specialist SpA selecting amongst two plans P1 and P2. Each plan has two tasks. A constraint in task T1 always fails.

```
***** AIR-CYL Air-cylinder Design System *****
*** Version date: (Dec 85)
*** Todays date:  (18 Feb 84)
*** User: DCBROWN

* Standard test/demo requirements to be used
*** Requirements Input Complete

--- Entering Specialist
    ...SpA... Mode = Design

!!! Note:
The specialist needs a plan.
Ask the sponsors for opinions
about each plan.

----- Entering Sponsor
      ...P1Sponsor...Plan = P1

----- Leaving Sponsor
      ....P1Sponsor...Result= SUITABLE

----- Entering Sponsor
      ...P2Sponsor...Plan = P2

----- Leaving Sponsor
      ....P2Sponsor...Result= PERFECT

!!! Note:
The opinion of the sponsors is that
there is a suitable plan and a perfect one.
The selector will pick one.

----- Entering Selector
      ...SpASelector

----- Leaving Selector
      ....SpASelector...Result= P2

!!! Note:
The selector's knowledge specifies to
pick P2 if it is perfect and hasnt been tried.

----- Entering Plan
      ...P2... Type = Design

------- Entering Task...T1

!!! Note:
Plan 2 uses Task 1 which uses
Constraint 1 which has been
fixed to always fail.

--------- Entering TEST-CONSTRAINTS...(C1)

--------- Leaving TEST-CONSTRAINTS....(C1)
          ...Result=
(Msg  Msg:MsgType       Failure
      Msg:MsgSubType    Constraint
      Msg:FromName      C1
      Msg:FromType      Constraint
      Msg:Message       "Constraint failure"
      Msg:Explanation   "C1 forced failure"
      Msg:InPlan        P2
      Msg:InMode        Design
          ......
)

------- Leaving Task....T1
      ...Result=

(Msg  Msg:MsgType Failure  .....)


----- Leaving Plan....P2
      ...Result=

(Msg  Msg:MsgType Failure  .....)


!!! Note:
These agents know nothing about
failure recovery. The task fails and
the plan failure follows.  A new plan
must be found.  Ask the sponsors.

----- Entering Sponsor

     ...P1Sponsor...Plan = P1
------ Leaving Sponsor
      ....P1Sponsor...Result= PERFECT

----- Entering Sponsor
      ...P2Sponsor...Plan = P2

----- Leaving Sponsor
      ....P2Sponsor...Result= RULE-OUT

!!! Note:
Note that both P1 and P2 have changed
their answers.  This is possible as the
situation has changed.

----- Entering Selector
      ...SpASelector

----- Leaving Selector
      ....SpASelector...Result= P1

!!! Note:
The selector picks the perfect plan.
```

```
----- Entering Plan...P1... Type = Design

------- Entering Task...T3

--------- Entering TEST-CONSTRAINTS...(C2)

--------- Leaving TEST-CONSTRAINTS....(C2)
        ...Result= Success Msg

------- Leaving Task....T3
      ...Result= Success Msg

------- Entering Task...T1

!!! Note:
C1 fails again, leading to
plan failure.

--------- Entering TEST-CONSTRAINTS...(C1)

--------- Leaving TEST-CONSTRAINTS....(C1)
        ...Result=

(Msg  Msg:MsgType Failure .....)


------- Leaving Task....T1
      ...Result=

(Msg  Msg:MsgType Failure ...... )


----- Leaving Plan....P3
      ...Result=

(Msg  Msg:MsgType Failure ..... )


!!! Note:
Ask the sponsors again,


----- Entering Sponsor
      ...P1Sponsor...Plan = P1

----- Leaving Sponsor
      ....P1Sponsor...Result= RULE-OUT

----- Entering Sponsor
      ...P2Sponsor...Plan = P2

----- Leaving Sponsor
      ....P2Sponsor...Result= RULE-OUT
```

```
!!! Note:
The selector gives up.

----- Entering Selector
      ...SpASelector

----- Leaving Selector....SpASelector
      ...Result= ApplicablePlanNotFound

!!! Note:
With no more plans to try
the specialist will fail.

--- Leaving Specialist....SpA
    ...Result=

(Msg  Msg:MsgType Failure ..... )


*** Design attempt fails
*** Version date: (Dec 85)
*** Todays date:  (18 Feb 84)
*** User: DCBROWN
***** AIR-CYL Air-cylinder Design System *****
```

6 REFERENCES

[BROW83]
Brown, D. C. and Chandrasekaran, B.,
An approach to expert systems for mechanical design,
In: IEEE Computer Society, Trends and Applications ´83,
May 1983, NBS, Gaithersburg, MD, pp. 173-180

[BROW84A]
Brown, D. C.,
Expert Systems for Design Problem-Solving using Design Refinement with Plan Selection and Redesign,
Unpublished Ph.D. Dissertation, Ohio State University,
August 1984, CIS Dept., OSU, Columbus, OH 43210

[BROW84B]
Brown, D. C. and Chandrasekaran, B.,
Expert Systems for a Class of Mechanical Design Activity,
In: Proceedings of the Working Conference on Knowledge Engineering in Computer-Aided Design,
IFIP WG 5.2, (Ed.) Gero, J.
September 1984, Budapest, Hungary

[BROW85]
Brown, D. C.,
Capturing Mechanical Design Knowledge,

In: Proceedings of the 1985 ASME International Computers in Engineering Conference,
August 1985, Boston, MA.

[BYLA83]
Bylander, T., Mittal, S. and Chandrasekaran, B.,
CSRL: A language for expert systems for diagnosis,
In: Proceedings of the 8th International Joint Conference on Artificial Intelligence,
Karlsruhe, W. Germany, August 1983, pp. 218-221

[CHAN83]
Chandrasekaran, B.,
Towards a taxonomy of problem-solving types,
AI Magazine, Vol. 4, No. 1, pp. 9-17, 1983

[FRIE79]
Friedland, P.
Knowledge-based experimental design in molecular genetics,
In: Proceedings of the 6th International Joint Conference on Artificial Intelligence,
IJCAI, Tokyo, August 1979, pp. 285-287.

[KOWA83]
Kowalski, T. and Thomas, D.,
The VLSI Design Automation Assistant: Prototype System,
In: Proceedings of the 20th Design Automation Conference,
IEEE, 1983, pp.479-483

[MCDE82]
McDermott, J.,
R1 -- a rule-based configurer of computer systems,
In: Artificial Intelligence, Vol. 19, No. 1,
Sept. 1982, pp. 39-88

[MOST83]
Mostow, J.,
Program Transformations for VLSI,
In: Proceedings of the 8th International Joint Conference on Artificial Intelligence,
IJCAI, Karlsruhe, August 1983, pp. 40-43.

[SIMO81]
Simon, H. A.,
The Sciences of the Artificial,
MIT Press, 1981,
1st Edition, 1969.

[STAL77]
Stallman, R. and Sussman, G. J.,
Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis,
In: Artificial Intelligence, Vol. 9, pp. 135-196, 1977.
Also in MIT AI Lab Memo. 380, 1976.

END

DTIC

9 — 86