CLASSIFICATION PROBLEM SOLVING

William J. Clancey
Heuristic Programming Project
Computer Science Department
Stanford University
Stanford, CA 94305

ABSTRACT

A broad range of heuristic programs—embracing forms of diagnosis, catalog selection, and skeletal planning—accomplish a kind of well-structured problem solving called classification. These programs have a characteristic inference structure that systematically relates data to a pre-enumerated set of solutions by abstraction, heuristic association, and refinement. This level of description specifies the knowledge needed to solve a problem, independent of its representation in a particular computer language. The classification problem-solving model provides a useful framework for recognizing and representing similar problems, for designing representation tools, and for understanding why non-classification problems require different problem-solving methods.*

I INTRODUCTION

Over the past decade a variety of heuristic programs have been written to solve problems in diverse areas of science, engineering, business, and medicine. Yet, presented with a given "knowledge engineering tool," such as EMYCIN (van Melle, 1979), we are still hard-pressed to say what kinds of problems it can be used to solve well. Various studies have demonstrated advantages of using one representation language instead of another—for ease in specifying knowledge relationships, control of reasoning, and perspicuity for maintenance and explanation (Clancey, 1981, Swartout, 1981, Aiello, 1983, Aikins, 1983, Clancey, 1983a). Other studies have characterized in low-level terms why a given problem might be inappropriate for a given language, for example, because data are time-varying or subproblems interact (Hayes-Roth et al., 1983). But attempts to describe *kinds of problems* in terms of shared features have not been entirely satisfactory: Applications-oriented descriptions like "diagnosis" are too general (e.g., the program might not use a device model), and technological terms like "rule-based" don't describe what problem is being solved (Hayes, 1977, Hayes, 1979). Logic has been suggested as a tool for a "knowledge level" analysis that would specify what a heuristic program does, independent of its implementation in a programming language (Nilsson, 1981, Newell, 1982). However, we have lacked a set of terms and relations for doing this.

In an attempt to specify in some canonical terms what many heuristic programs known as "expert systems" do, an analysis was made of ten rule-based systems (including MYCIN, SACON, and The Drilling Advisor), a frame-based system (GRUNDY) and a program coded directly in LISP (SOPHIE III). There is a striking pattern: These programs proceed through easily identifiable phases of data abstraction, heuristic mapping onto a hierarchy of pre-enumerated solutions, and refinement within this hierarchy. In short, these programs do what is commonly called *classification.*

Focusing on content rather than representational technology, this paper proposes a set of terms and relations for describing the knowledge used to solve a problem by classification. Subsequent sections describe and illustrate the classification model in the analysis of MYCIN, SACON, GRUNDY, and SOPHIE III. Significantly, a knowledge level description of these programs corresponds very well to psychological models of expert problem solving. This suggests that the classification problem solving model captures general principles of how experiential knowledge is organized and used, and thus generalizes some cognitive science results. There are several strong implications for the practice of building expert systems and continued research in this field.

II CLASSIFICATION PROBLEM SOLVING DEFINED

We develop the idea of classification problem solving by starting with the common sense notion and relating it to the reasoning that occurs in heuristic programs.

A. Simple classification

As the name suggests, the simplest kind of classification problem is to identify some unknown object or phenomenon as a member of a known class of objects or phenomena. Typically, these classes are stereotypes that are hierarchically organized, and the process of identification is one of matching observations of an unknown entity against features of known classes. A paradigmatic example is identification of a plant or animal, using a guidebook of features, such as coloration, structure, and size.

Some terminology we will find helpful: The *problem* is the object or phenomenon to be identified; *data* are observations describing this problem; possible *solutions* are patterns (variously called schema, frames or units); each solution has a set of *features* (slots or facets) that in some sense describe the concept either categorically or probabilistically; solutions are grouped into *a specialization hierarchy* based on their features (in general, not a single hierarchy, but multiple, directed acyclic graphs); a *hypothesis* is a solution that is under consideration; *evidence* is data that partially matches some hypothesis; the *output* is some solution.

The essential characteristic of a classification problem is that the problem solver selects from a set of pre-enumerated solutions. This does not mean, of course, that the "right answer" is necessarily one of these solutions, just that the problem solver will only attempt to match the data against the known solutions, rather than construct a new one. Evidence can be uncertain and matches partial, so the output might be a ranked list of hypotheses.

Besides matching, there are several *rules of inference* for making assertions about solutions. For example, evidence for a class is indirect evidence that one of its subtypes is present. Conversely, given a closed world assumption, evidence against all of the subtypes is evidence against a class. *Search operators* for finding a solution also capitalize on the hierarchical structure of the solution space. These operators

*This research has been supported in part by ONR and ARI Contract N00014-79C-0302. Computational resources have been provided by the SUMEX-AIM facility (NIH grant RR00785). Many of the ideas presented here were stimulated by discussions with Denny Brown in our attempt to develop a framework for teaching knowledge engineering. I am also grateful to Tom Dietterich, Steve Hardy, and Peter Szolovits for their suggestions and encouragement. The Drilling Advisor mentioned herein is a product of Teknowledge, Inc.

include: *refining* a hypothesis to a more specific classification; *categorizing* the problem by considering superclasses of partially matched hypotheses; and *discriminating* among hypotheses by contrasting their superclasses (Patil, 1981, Pople, 1982, Clancey, 1984). For simplicity, we will refer to the entire process of applying these rules of inference and operators as *refinement*. The specification of this process—a control strategy—is an orthogonal issue which we will consider later.

B. Data abstraction

In the simplest problems, data are solution features, so the matching and refining process is direct. For example, an unknown organism in MYCIN can be classified directly given the supplied data of gram stain and morphology.

For many problems, solution features are not supplied as data, but are inferred by *data abstraction*. There are three basic relations for abstracting data in heuristic programs:

- *qualitative abstraction* of quantitative data ("if the patient is an adult and white blood count is less than 2500, then the white blood count is low");
- *definitional abstraction* ("if the structure is one-dimensional of network construction, then its shape is a beam"); and
- *generalization* in a subtype hierarchy ("if the client is a judge, then he is an educated person").

These interpretations are usually made by the program with certainty; thresholds and qualifying contexts are chosen so the conclusion is categorical. It is common to refer to this knowledge as "descriptive," "factual," or "definitional."

C. Heuristic classification

In simple classification, the data may directly match the solution features or may match after being abstracted. In heuristic classification, solution features may also be matched heuristically. For example, MYCIN does more than identify an unknown organism in terms of features of a laboratory culture: It heuristically relates an abstract characterization of the patient to a classification of diseases. We show this *inference structure* schematically, followed by an example (Figure 1).

Basic observations about the patient are abstracted to patient categories, which are heuristically linked to diseases and disease categories. While only a subtype link with E.coli infection is shown here, evidence may actually derive from a combination of inferences. Some data might directly match E.coli by identification. Discrimination with competing subtypes of gram-negative infection might also provide evidence. As stated earlier, the order in which these inferences are made is a matter of control strategy.

The important link we have added is a heuristic association between a characterization of the patient ("compromised host") and categories of diseases ("gram-negative infection"). Unlike the factual and hierarchical evidence propagation we have considered to this point, this inference makes a great leap. A *heuristic relation* is based on some implicit, possibly incomplete, model of the world. This relation is often empirical, based just on experience; it corresponds most closely to the "rules of thumb" often associated with heuristic programs (Feigenbaum, 1977).

Heuristics of this type reduce search by skipping over intermediate relations (this is why we don't call abstraction relations "heuristics"). These associations are usually uncertain because the intermediate relations may not hold in the specific case. Intermediate relations may be omitted because they are unobservable or poorly understood. In a medical diagnosis program, heuristics typically skip over the causal relations between symptoms and diseases.

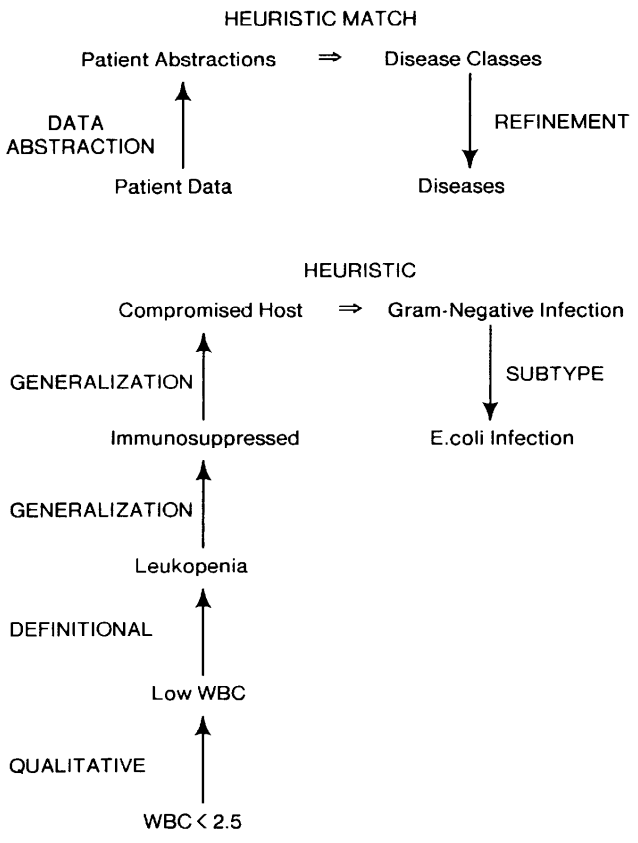


Figure 1: Inference structure of MYCIN

To repeat, classification problem solving involves heuristic association of an abstracted problem statement onto features that characterize a solution. This can be shown schematically in simple terms (Figure 2).

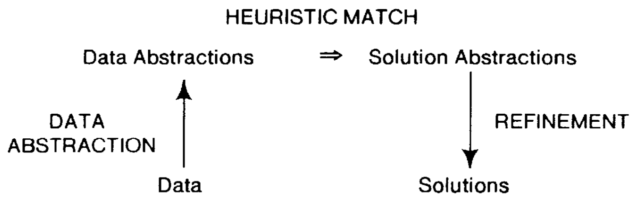


Figure 2: Classification problem solving inference structure

This diagram summarizes how a distinguished set of terms (data, data abstractions, solution abstractions, and solutions) are related systematically by *different* kinds of relations and rules of inference. This is the *structure of inference* in classification problem solving.

In a study of physics problem solving, Chi (Chi, et al., 1981) calls data abstractions "transformed" or "second order problem features." In an important and apparently common variant of the simple model, data abstractions are themselves patterns that are heuristically matched. In essence, there is a sequence of classification problems. GRUNDY, analyzed below, illustrates this.

D. Search in classification problem solving

The issue of search is orthogonal to the kinds of inference we have been considering. "Search" refers to how a network made up of abstraction, heuristic, and refinement relations is interpreted, how the flow of inference actually might proceed in solving a problem. Following Hayes (Hayes, 1977), we call this the *process structure*. There are three basic process structures in classification problem solving:

1. *Data-directed search*: The program works forwards from data to abstractions, matching solutions until all possible (or non-redundant) inferences have been made.

2. *Solution- or Hypothesis-directed search*: The program works backwards from solutions, collecting evidence to support them, working backwards through the heuristic relations to the data abstractions and required data to solve the problem. If solutions are hierarchically organized, then categories are considered before direct features of more specific solutions.

3. *Opportunistic search*: The program combines data and hypothesis-directed reasoning (Hayes-Roth and Hayes-Roth, 1979). Data abstraction rules tend to be applied immediately as data become available. Heuristic rules "trigger" hypotheses, followed by a focused, hypothesis-directed search. New data may cause refocusing. By reasoning about solution classes, search need not be exhaustive.

Data- and hypothesis-directed search are not to be confused with the implementation terms "forward" or "backward chaining." R1 provides a superb example of how different implementation and knowledge level descriptions can be. Its rules are *interpreted* by forward-chaining, but it does a form of hypothesis-directed search, systematically setting up subproblems by a fixed procedure that focuses reasoning on spatial subcomponents of a solution (McDermott, 1982).

The degree to which search is *focused* depends on the level of indexing in the implementation and how it is exploited. For example, MYCIN's "goals" are solution classes (e.g., types of bacterial meningitis), but selection of rules for specific solutions (e.g., E.coli meningitis) is unordered. Thus, MYCIN's search within each class is unfocused (Clancey, 1983b).

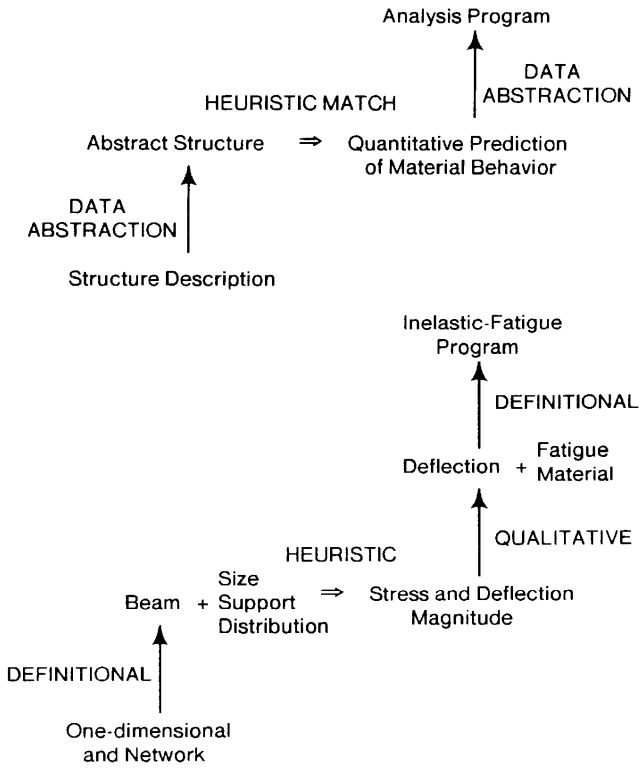
The choice of process structure depends on the number of solutions, whether they can be categorically constrained, usefulness of data (the density of rows in a data/solution matrix), and the cost for acquiring data.

III EXAMPLES OF CLASSIFICATION PROBLEM SOLVING

Here we schematically describe the architectures of SACON, GRUNDY, and SOPHIE III in terms of classification problem solving. These are necessarily very brief descriptions, but reveal the value of this kind of analysis by helping us to understand what the programs do. After a statement of the problem, the general inference structure and an example inference path are given, followed by a brief discussion.

A. SACON

Problem: SACON (Bennett, et al., 1978) selects classes of behavior that should be further investigated by a structural analysis simulation program (Figure 3).



Analysis Program
DATA ABSTRACTION
HEURISTIC MATCH
Abstract Structure
Quantitative Prediction of Material Behavior
DATA ABSTRACTION
Structure Description
Inelastic-Fatigue Program
DEFINITIONAL
Deflection + Fatigue Material
QUALITATIVE
HEURISTIC
Beam + Size Support Distribution
Stress and Deflection Magnitude
DEFINITIONAL
One-dimensional and Network

Figure 3: Inference structure of SACON

Discussion: SACON solves two problems by classification—analyzing a structure and then selecting a program. It begins by heuristically selecting a simple numeric model for analyzing a structure (such as an airplane wing). The model produces stress and deflection estimates, which the program then abstracts in terms of features that hierarchically describe different configurations of the MARC simulation program. There is no refinement because the solutions to the first problem are just a simple set of possible models, and the second problem is only solved to the point of specifying program classes. (In another software configuration system we analyzed, specific program input parameters are inferred in a refinement step.)

B. GRUNDY

Problem: GRUNDY (Rich, 1979) heuristically classifies a reader's personality and selects books he might like to read (Figure 4).

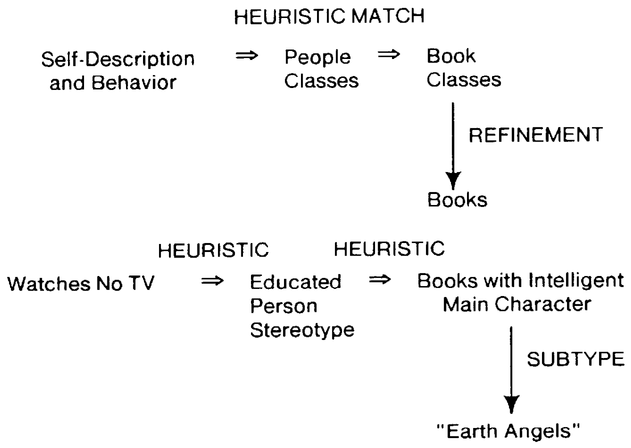


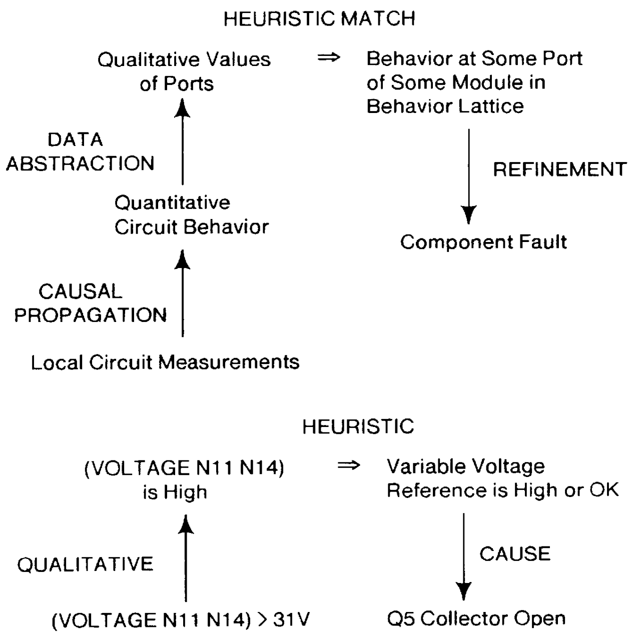
Figure 4: Inference structure of GRUNDY

Discussion: GRUNDY solves two classification problems heuristically. Illustrating the power of a knowledge level analysis, we discover that the people and book classifications are not distinct in the implementation. For example, "fast plots" is a book characteristic, but in the implementation "likes fast plots" is associated with a person stereotype. The relation between a person stereotype and "fast plots" is heuristic and should be distinguished from abstractions of people and books. One objective of the program is to learn better people stereotypes (user models). The classification description of the user modeling problem shows that GRUNDY should also be learning better ways to characterize books, as well as improving its heuristics. If these are not treated separately, learning may be hindered. This example illustrates why a knowledge level analysis should precede representation.

It is interesting to note that GRUNDY does not attempt to perfect the user model before recommending a book. Rather, refinement of the person stereotype occurs when the reader rejects book suggestions. Analysis of other programs indicates that this multiple-pass process structure is common. For example, the Drilling Advisor makes two passes on the causes of sticking, considering general, inexpensive data first, just as medical programs commonly consider the "history and physical" before laboratory data.

C. SOPHIE III

Problem: SOPHIE III (Brown, et al., 1982) classifies an electronic circuit in terms of the component that is causing faulty behavior (Figure 5).



HEURISTIC MATCH
Qualitative Values of Ports
⇒
Behavior at Some Port of Some Module in Behavior Lattice
DATA ABSTRACTION
REFINEMENT
Quantitative Circuit Behavior
Component Fault
CAUSAL PROPAGATION
Local Circuit Measurements
HEURISTIC
(VOLTAGE N11 N14) is High
⇒
Variable Voltage Reference is High or OK
QUALITATIVE
CAUSE
(VOLTAGE N11 N14) > 31V
Q5 Collector Open

Figure 5: Inference structure of SOPHIE

Discussion: SOPHIE's set of pre-enumerated solutions is a lattice of valid and faulty circuit behaviors. In contrast with MYCIN, solutions are device states and component flaws, not stereotypes of disorders, and they are related causally, not by features. Data are not just external device behaviors, but include internal component measurements propagated by the causal analysis of the LOCAL program. Reasoning about assumptions plays a central role in matching hypotheses. In spite of these differences, the inference structure of abstractions, heuristic relations, and refinement fits the classification model, demonstrating its generality and usefulness for describing complex reasoning.

IV CAUSAL PROCESS CLASSIFICATION

To further illustrate the value of a knowledge level analysis, we describe a generic inference structure common to medical diagnosis programs, which we call *causal process classification*, and use it to contrast the goals of electronic circuit and medical diagnosis programs.

In SOPHIE, valid and abnormal device states are exhaustively enumerated, can be directly confirmed, and are causally related to component failures. None of this is generally possible in medical diagnosis, nor is diagnosis in terms of component failures alone sufficient for selecting therapy. Medical programs that deal with multiple disease processes (unlike MYCIN) do reason about abnormal states (called *pathophysiologic states*, e.g., "increased pressure in the brain"), directly analogous to the abnormal states in SOPHIE. But curing an illness generally involves determining the cause of the component failure. These "final causes" (called diseases, syndromes, etiologies) are processes that affect the normal functioning of the body (e.g., trauma, infection, toxic exposure, psychological disorder). Thus, medical diagnosis more closely resembles the task of computer system diagnosis in considering how the body relates to its environment (Lane, 1980). In short, there are two problems: First to explain symptoms in terms of abnormal internal states, and second to explain this behavior in terms of external influences (as well as congenital and degenerative component flaws). This is the inference structure of programs like CASNET (Weiss, et al., 1978) and NEOMYCIN (Clancey, 1981) (Figure 6).

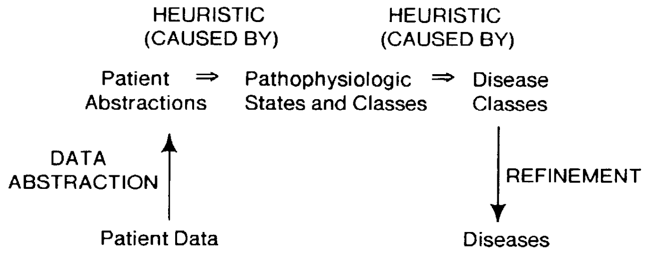


Figure 6: Inference structure of causal process classification

A network of causally related pathophysiologic states causally relates data to diseases**. The causal relations are themselves heuristic because they assume certain physiologic structure and behavior, which is often poorly understood and not represented. In contrast with pathophysiologic states, diseases are abstractions of *processes*--causal stories with agents, locations, and sequences of events. Disease networks are organized by these process features (e.g., an organ system taxonomy organizes diseases by location). A more general term for disease is *disorder stereotype*. In *process control* problems, such as chemical manufacturing, the most general disorder stereotypes correspond to stages in a process (e.g., mixing, chemical reaction, filtering, packaging). Subtypes correspond to what can go wrong at each stage (Clancey, 1984).

**Programs differ in whether they treat pathophysiologic states as independent solutions (NEOMYCIN) or find the causal path that best accounts for the data (CASNET). Moreover, a causal explanation of the data requires finding a state network, including normal states, that is internally consistent on multiple levels of detail. Combinatorial problems, as well as elegance, argue against pre-enumerating solutions, so such a network must be constructed, as in ABEL (Patil, 1981). In SOPHIE, the LOCAL program deals with most of the state interactions at the component level, others are captured in the exhaustive hierarchy of module behaviors. A more general solution is to use a structure/function device model and general diagnostic operators, as in DART (Genesereth, 1982).

To summarize, a knowledge level analysis reveals that medical and electronic diagnosis programs are not all trying to solve the same kind of problem. Examining the nature of solutions, we see that in a electronic circuit diagnosis program like SOPHIE solutions are component flaws. Medical diagnosis programs like CASNET attempt a second step, *causal process classification*, which is to explain abnormal states and flaws in terms of processes external to the device or developmental processes affecting its structure. It is this experiential knowledge—what can affect the device in the world—that is captured in disease stereotypes. This knowledge can't simply be replaced by a model of device structure and function, which is concerned with a different level of analysis.

V WHAT IS NON-CLASSIFICATION PROBLEM SOLVING?

We first summarize the applications we have considered by observing that all classification problem solving involves *selection* of a solution. We can characterize kinds of problems by what is being selected:

- *diagnosis*: solutions are faulty components (SOPHIE) or processes affecting the device (MYCIN);
- *user model*: solutions are people stereotypes in terms of their goals and beliefs (first phase of GRUNDY);
- *catalog selection*: solutions are products, services, or activities, e.g., books, personal computers, careers, travel tours, wines, investments (second phase of GRUNDY);
- *theoretical analysis*: solutions are numeric models (first phase of SACON);
- *skeletal planning*: solutions are plans, such as packaged sequences of programs and parameters for running them (second phase of SACON, also first phase of experiment planning in MOLGEN (Friedland, 1979)).

A common misconception is that the description "classification problem" is an inherent property of a problem, opposing, for example, classification with design (Sowa, 1984). However, classification problem solving, as defined here, is *a description of how a problem is solved.* If the problem solver has a priori knowledge of solutions and can relate them to the problem description by data abstraction, heuristic association, and refinement, then the problem can be solved by classification. For example, if it were practical to enumerate all of the computer configurations R1 might select, or if the solutions were restricted to a predetermined set of designs, the program could be reconfigured to solve its problem by classification.

Furthermore, as illustrated by ABEL, it is incorrect to say that medical diagnosis is a "classification problem." Only *routine* medical diagnosis problems can be solved by classification (Pople, 1982). When there are multiple, interacting diseases, there are too many possible combinations for the problem solver to have considered them all before. Just as ABEL reasons about interacting states, the physician must construct a consistent network of interacting diseases to explain the symptoms. The problem solver *formulates a solution*; he doesn't just make yes-no decisions from a set of fixed alternatives. For this reason, Pople calls non-routine medical diagnosis an ill-structured problem (Simon, 1973) (though it may be more appropriate to reserve this term for the theory formation task of the physician-scientist who is defining new diseases).

In summary, a useful distinction is whether a solution is selected or constructed. To *select* a solution, the problem solver needs experiential ("expert") knowledge in the form of *patterns of problems and solutions* and heuristics relating them. To *construct* a solution, the problem solver applies models of structure and behavior, by which objects can be assembled, diagnosed, or employed in some plan.

Whether the solution is taken off the shelf or is pieced together has important computational implications for choosing a representation. In particular, construction problem-solving methods such as constraint propagation and dependency-directed backtracking have data structure requirements that may not be easily satisfied by a given representation language. For example—returning to a question posed in the introduction—applications of EMYCIN are generally restricted to problems that can be solved by classification.

VI KNOWLEDGE LEVEL ANALYSIS

As a set of terms and relations for describing knowledge (e.g, data, solutions, kinds of abstraction, refinement operators, the meaning of "heuristic"), the classification model provides a *knowledge level analysis* of programs, as defined by Newell (Newell, 1982). It "serves as a specification of what a reasoning system should be able to do." Like a specification of a conventional program, this description is distinct from the representational technology used to implement the reasoning system. Newell cites Schank's conceptual dependency structure as an example of a knowledge level analysis. It indicates "what knowledge is required to solve a problem... how to encode knowledge of the world in a representation."

After a decade of "explicitly" representing knowledge in AI languages, it is ironic that the pattern of classification problems should have been so difficult to see. In retrospect, certain views were emphasized at the expense of others:

- *Procedureless languages.* In an attempt to distinguish heuristic programming from traditional programming, procedural constructs are left out of representation languages (such as EMYCIN, OPS, KRL (Lehnert and Wilks, 1979)). Thus, inference relations cannot be stated separately from how they are to be used (Hayes, 1977, Hayes, 1979).
- *Heuristic nature of problem solving.* Heuristic association has been emphasized at the expense of the relations used in data abstraction and refinement. In fact, some expert systems do only simple classification; they have no heuristics or "rules of thumb," the key idea that is supposed distinguish this class of computer programs.
- *Implementation terminology.* In emphasizing new implementation technology, terms such as "modular" and "goal directed" were more important to highlight than the content of the programs. In fact, "goal directed" characterizes any rational system and says very little about how knowledge is used to solve a problem. "Modularity" is a representational issue of indexing.

Nilsson has proposed that logic should be the *lingua franca* for knowledge level analysis (Nilsson, 1981). Our experience with the classification model suggests that the value of using logic is in adopting a set of terms and relations for describing knowledge (e.g., kinds of abstraction). Logic is valuable as a tool for knowledge level analysis because it emphasizes *relations*, not just implication.

While rule-based languages do not make important knowledge level distinctions, they have nevertheless provided an extremely successful programming framework for classification problem solving. Working backwards (backchaining) from a pre-enumerated set of solutions guarantees that only the relevant rules are tried and useful data considered. Moreover, the program designer is encouraged to use means-ends analysis, a clear framework for organizing rule writing.

VII RELATED ANALYSES

Several researchers have described portions of the classification problem solving model, influencing this analysis. For example, in CRYSALIS (Engelmore and Terry, 1979) data and hypothesis abstraction are clearly separated. The EXPERT rule language (Weiss, 1979) similarly distinguishes between "findings" and a taxonomy of hypotheses. In PROSPECTOR (Hart, 1977), rules are expressed in terms of relations in a semantic network. In CENTAUR (Aikins, 1983), a variant of MYCIN, solutions are explicitly *prototypes* of diseases. Chandrasekaran and his associates have been strong proponents of the classification model: "The normal problem-solving activity of the physician... (is) a process of classifying the case as an element of a disease taxonomy" (Chandrasekaran and Mittal, 1983). Recently, Chandrasekaran and Weiss and Kulikowski have generalized the classification schemes used by their programs (MDX and EXPERT) to characterize problems solved by other expert systems (Chandrasekaran, 1984, Weiss and Kulikowski, 1984).

A series of knowledge representation languages beginning with KRL have identified structured abstraction and matching as a central part of problem solving (Bobrow and Winograd, 1979). Building on the idea that "frames" are not just a computational construct, but a theory about a kind of knowledge (Hayes, 1979), cognitive science studies have described problem solving in terms of classification. For example, routine physics problem solving is described by Chi (Chi, et al., 1981) as a process of data abstraction and heuristic mapping onto solution schemas ("experts cite the abstracted features as the relevant cues (of physics principles)"). The inference structure of SACON, heuristically relating structural abstractions to numeric models, is the same.

Related to the physics problem solving analysis is a very large body of research on the nature of schemas and their role in understanding (Schank, 1975, Rumelhart and Norman, 1983). More generally, the study of classification, particularly of objects, also called *categorization*, has been a basic topic in psychology for several decades (e.g., see the chapter on "conceptual thinking" in (Johnson-Laird and Wason, 1977)). However, in psychology the emphasis has been on the nature of categories and how they are formed (an issue of learning). The programs we have considered make an identification or selection from a pre-existing classification (an issue of memory retrieval). In recent work, Kolodner combines the retrieval and learning process in an expert system that learns from experience (Kolodner, 1982). Her program uses the MOPS representation, a classification model of memory that interleaves generalizations with specific facts (Kolodner, 1983).

VIII CONCLUSIONS

A wide variety of problems can be described in terms of heuristic mapping of data abstractions onto a fixed, hierarchical network of solutions. This problem solving model is supported by psychological studies of human memory and the role of classification in understanding. There are significant implications for expert systems research:

- The model provides a *high-level structure for decomposing problems*, making it easier to recognize and represent similar problems. For example, problems can be characterized in terms of sequences of classification problems. Catalog selection programs might be improved by incorporating a more distinct phase of user modelling, in which needs or requirements are classified first. Diagnosis programs might profitably make a stronger separation between device-history stereotypes and disorder knowledge. A generic knowledge engineering tool can be designed specifically for classification problem solving. The advantages for knowledge acquisition carry over into explanation and teaching.
- The model provides a *basis for choosing application problems*. For example, problems can be selected that will teach us more about the nature of abstraction and how other forms of inference (e.g., analogy, simulation, constraint posting) are combined with classification.
- The model provides a *foundation for describing representation languages* in terms of epistemologic adequacy (McCarthy and Hayes, 1969), so that the leverage they provide can be better understood. For example, for classification it is advantageous for a language to provide constructs for representing problem solutions as a network of schemas.
- The model provides a *focus for cognitive studies* of human categorization of knowledge and search strategies for retrieval and matching, suggesting principles that might be used in expert programs. Learning research might similarly focus on the inference and process structure of classification problem solving.

Finally, it is important to remember that expert systems are programs. Basic computational ideas such as input, output, and sequence, are essential for describing what they do. The basic methodology of our study has been to ask, "What does the program conclude about? How does it get there from its input?" We characterize the flow of inference, identifying data abstractions, heuristics, implicit models and assumptions, and solution categories along the way. If heuristic programming is to be different from traditional programming, a knowledge level analysis should always be pursued to the deepest levels of our understanding, even if practical constraints prevent making explicit in the implemented program everything that we know. In this way, knowledge *engineering* can be based on sound principles that unite it with studies of cognition and representation.

References

Aiello, N. *A comparative study of control strategies for expert systems: AGE implementation of three variations of PUFF*, in *Proceedings of the National Conference on AI*, pages 1-4, Washington, D.C., August, 1983.

Aikins J. S. Prototypical knowledge for expert systems. *Artificial Intelligence*, 1983, *20(2)*, 163-210.

Bennett, J., Creary, L., Englemore, R., and Melosh, R. *SACON: A knowledge-based consultant for structural analysis.* STAN-CS-78-699 and HPP Memo 78-23, Stanford University, Sept 1978.

Bobrow, D. and Winograd, T. KRL: Another perspective. *Cognitive Science*, 1979, *3*, 29-42.

Brown, J. S., Burton, R. R., and de Kleer, J. Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman and J. S. Brown (editors), *Intelligent Tutoring Systems*, pages 227-282. Academic Press, 1982.

Chandrasekaran, B. Expert systems: Matching techniques to tasks. In W. Reitman (editor), *AI Applications for Business*, pages 116-132. Ablex Publishing Corp., 1984.

Chandrasekaran, B. and Mittal, S. Conceptual representation of medical knowledge. In M. Yovits (editor), *Advances in Computers*, pages 217-293. Academic Press, New York, 1983.

Chi, M. T. H., Feltovich, P. J., Glaser, R. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 1981, *5*, 121-152.

Clancey, W. J. and Letsinger, R. *NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching*, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 829-836, August, 1981. (Revised version to appear in Clancey and Shortliffe (editors), *Readings in Medical Artificial Intelligence: The First Decade*, Addison-Wesley, 1983).

Clancey, W. J. *The advantages of abstract control knowledge in expert system design*, in *Proceedings of the National Conference on AI*, pages 74-78, Washington, D.C., August, 1983.

Clancey, W. J. The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence*, 1983, *20(3)*, 215-251.

Clancey, W. J. *Acquiring, representing, and evaluating a competence model of diagnosis*. HPP Memo 84-2, Stanford University, February 1984. (To appear in Chi, Glaser, and Farr (Eds.), *The Nature of Expertise*, in preparation.).

Engelmore, R. and Terry, A. *Structure and function of the CRYSALIS system*, in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 250-256, August, 1979.

Feigenbaum, E. A. *The art of artificial intelligence: I. Themes and case studies of knowledge engineering*, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1014-1029, August, 1977.

Friedland, P. E. *Knowledge-based experiment design in molecular genetics*. Technical Report STAN-CS-79-771, Stanford University, October 1979.

Genesereth, M. R. *Diagnosis using hierarchical design models*, in *Proceedings of the National Conference on AI*, pages 278-283, Pittsburgh, PA, August, 1982.

Hart, P. E. *Observations on the development of expert knowledge-based systems*, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1001-1003, August, 1977.

Hayes, P.J. *In defence of logic*, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 559-565, August, 1977.

Hayes, P. The logic of frames. In D. Metzing (editor), *Frame Conceptions and Text Understanding*, pages 45-61. de Gruyter, 1979.

Hayes-Roth, B. and Hayes-Roth, F. A cognitive model of planning. *Cognitive Science*, 1979, *3*, 275-310.

Hayes-Roth, F., Waterman, D., and Lenat, D. (eds.). *Building expert systems*. New York: Addison-Wesley 1983.

Johnson-Laird, P. N. and Wason, P. C. *Thinking: Readings in Cognitive Science*. Cambridge: Cambridge University Press 1977.

Kolodner, J. L. *The role of experience in development of expertise*, in *Proceedings of the National Conference on AI*, pages 273-277, Pittsburgh, PA, August, 1982.

Kolodner, J. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 1983, *7*, 243-280.

Lane, W. G. Input/output processing. In Stone, H. S. (editor), *Introduction to Computer Architecture, 2nd Edition*, chapter 6. Science Research Associates, Inc., Chicago, 1980.

Lehnert, W., and Wilks, Y. A critical perspective on KRL. *Cognitive Science*, 1979, *3*, 1-28.

McCarthy, J. and Hayes, P. Some philosophical problems from the standpoint of Artificial Intelligence. In B. Meltzer and D. Michie (editors), *Machine Intelligence 4*, pages 463-502. Edinburgh University Press, 1969.

McDermott, J. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 1982, *19(1)*, 39-88.

Newell, A. The knowledge level. *Artificial Intelligence*, 1982, *18(1)*, 87-127.

Nilsson, N. J. The interplay between theoretical and experimental methods in Artificial Intelligence. *Cognition and Brain Theory*, 1981, *4(1)*, 69-74.

Patil, R. S., Szolovits, P., and Schwartz, W. B. *Causal understanding of patient illness in medical diagnosis*, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 893-899, August, 1981.

Pople, H. Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnostics. In P. Szolovits (editor), *Artificial Intelligence in Medicine*, pages 119-190. Westview Press, 1982.

Rich, E. User modeling via stereotypes. *Cognitive Science*, 1979, *3*, 355-366.

Rumelhart, D. E. and Norman, D. A. *Representation in memory*. Technical Report CHIP-116, Center for Human Information Processing, University of California, June 1983.

Schank, R. C., and Abelson, R. P. *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates 1975.

Simon, H. A. The structure of ill structured problems. *Artificial Intelligence*, 1973, *4*, 181-201.

Sowa, J. F. *Conceptual Structures*. Reading, MA: Addison-Wesley 1984.

Swartout W. R. *Explaining and justifying in expert consulting programs*, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 815-823, August, 1981.

van Melle, W. *A domain-independent production rule system for consultation programs*, in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 923-925, August, 1979.

Weiss, S. M. and Kulikowski, C. A. *EXPERT: A system for developing consultation models*, in *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 942-947, August, 1979.

Weiss, S. M. and Kulikowski, C. A. *A Practical Guide to Designing Expert Systems*. Totowa, NJ: Rowman and Allanheld 1984.

Weiss, S. M., Kulikowski, C. A., Amarel, S., and Safir, A. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 1978, *11*, 145-172.