January 1987

Report No. STAN-CS-87-1183
Also numbered KSL-87-12

The Knowledge Engineer as Student: Metacognitive bases for asking good questions

by

William J. Clancey

Department of Computer Science

Stanford University
Stanford, CA 94305

The Knowledge Engineer as Student:

Metacognitive bases for asking good questions

by

William J. Clancey

Stanford Knowledge Systems Laboratory
Department of Computer Science
701 Welch Road, Building C
Palo Alto, CA 94304

The studies reported here were supported (in part) by:

The Office of Naval Research
Personnel and Training Research Programs
Psychological Sciences Division
Contract No. N00014-85K-0305

The Josiah Macy, Jr. Foundation
Grant No. B852005
New York City

The views and conclusions contained in this document are author's and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Office of Naval Research or the U.S. Government.

Table of Contents

Abstract

Knowledge engineers are efficient, active learners. They systematically approach new domains and acquire knowledge to solve routine, practical problems. By modeling their methods, we may develop a basis for teaching other students how to direct their own learning. In particular, a knowledge engineer is good at detecting gaps in a knowledge base and asking focused questions to improve an expert system's performance. This ability stems from domain-general knowledge about: problem-solving procedures, the categorization of routine problem-solving knowledge, and domain and task differences. This paper studies these different forms of metaknowledge, and illustrates its incorporation in an intelligent tutoring system. A model of learning is presented that describes how the knowledge engineer detects problem-solving failures and tracks them back to gaps in domain knowledge, which are then reformulated as questions to ask a teacher. We describe how this model of active learning is being developed and tested in a knowledge acquisition program for an expert system.

1. Introduction

A knowledge engineer can be viewed as a special kind of student. Her goal is to develop computational models of complex problem solving by watching and questioning an expert and incrementally testing her model on a set of selected problem cases? Characteristically, the knowledge engineer (KE) is in complete control of this process. Her construction of a problem-solving model is almost completely self-directed; she is an active learner. The KE thus provides us with an excellent basis for studying methods that any student might use for approaching new problem domains and acquiring the knowledge to solve a set of practical problems.

Although there is some self-selection among KEs, so that people who are naturally quick learners are attracted to this profession (and there are some dilettantes), the knowledge engineering process is a skill that can be taught. In essence a knowledge engineer learns how to ask good questions by learning useful representations of knowledge, and by practicing the art of directing an expert to teach her what she needs to know. The activity of incrementally improving a computational problem-solving model (the expert system) on a well-defined sequence of cases focuses the learning activity. An intelligent tutoring system focuses learning in a similar way by engaging a student in case-method dialogues. Can we teach a student to play an active role in directing the tutoring program during these dialogues, in the same way a

---

'In this paper we use feminine pronouns to refer to KEs, though there are as many men in the profession; for symmetry we refer to students with masculine pronouns.

knowledge engineer directs her teacher?

This paper studies the knowledge-acquisition process by reviewing a variety of KE interview and knowledge-base critiquing heuristics (Section 3). Generalizing from these examples, we show how learning heuristics are intimately related to and derived from particular knowledge representation languages (presented as an introductory framework in Section 2). Finally, we consider how the general model of learning that emerges can be formalized in a knowledge acquisition program (Section 4.1) and then used as a standard for interpreting and guiding a student's behavior (Section 4.2). Relation to current work in machine learning and philosophical problems are considered in the final sections of the paper.

1.1. Basing teaching on a model of learning

It is generally accepted that development of teaching programs should proceed from a model of the learning process. One approach is to design a teaching program so that it encourages the student to improve his understanding, such as by making predictions about some phenomenon and formulating experiments to test them (Crovello and McDaniel, shed). Although most computer-aided instruction programs of this type provide the student with a simulation of a physical process (e.g., an electronic circuit), artificial intelligence (AI) programming techniques enable us to provide a model of a problem-solving process as well. In particular, an expert system can be presented as an object of study, as in GUIDON (Clancey, 1982, Clancey, 1987).

In our previous research we have developed methods by which a model of the diagnostic process can be explored by a student (Richer and Clancey, 1985). This program, called GUIDON-WATCH, is designed to facilitate understanding the knowledge organization and diagnostic strategy of the underlying expert system, NEOMYCIN (Clancey and Letsinger, 1984), presented as a model for the student to study and emulate. A window-menu system for browsing a knowledge base overprints taxonomies and tables to show the flow and history of reasoning. Experience shows that GUIDON-WATCH is quite useful for a knowledge engineer debugging NEOMYCIN and for short lecture-style demonstrations to students and other researchers (e.g., using a blinking display to show the strategy of "looking up" and "looking down" through disease categories). However, we have not formalized or built into the program what a student using GUIDON-WATCH should be trying to do. While we have reified the process of diagnosis-making it concrete so it can be studied-we have not made explicit the *goal* structure of a student who is studying the program. Specifically, what is the learning process involved in studying and understanding a model of problem solving, in this case an expert system?

We are already familiar with the process of learning by studying an expert problem

solver-this is what a KE does. The symmetry is shown in Figure l-l: The KE actively probes the expert, **listening** to and organizing explanations in order to improve her model of problem solving, the expert system. **By analogy,** a student actively probes a computer tutor, listening to and organizing explanations in order to improve his own problem-solving performance. Our thesis is that by studying and modeling what a KE does, we will be able to formulate a model of learning that can be incorporated in the design of a computer tutor. In particular we are interested in modeling the learning process involved in interacting with an expert-teacher, in order to replicate his behavior in some well-defined problem domain.

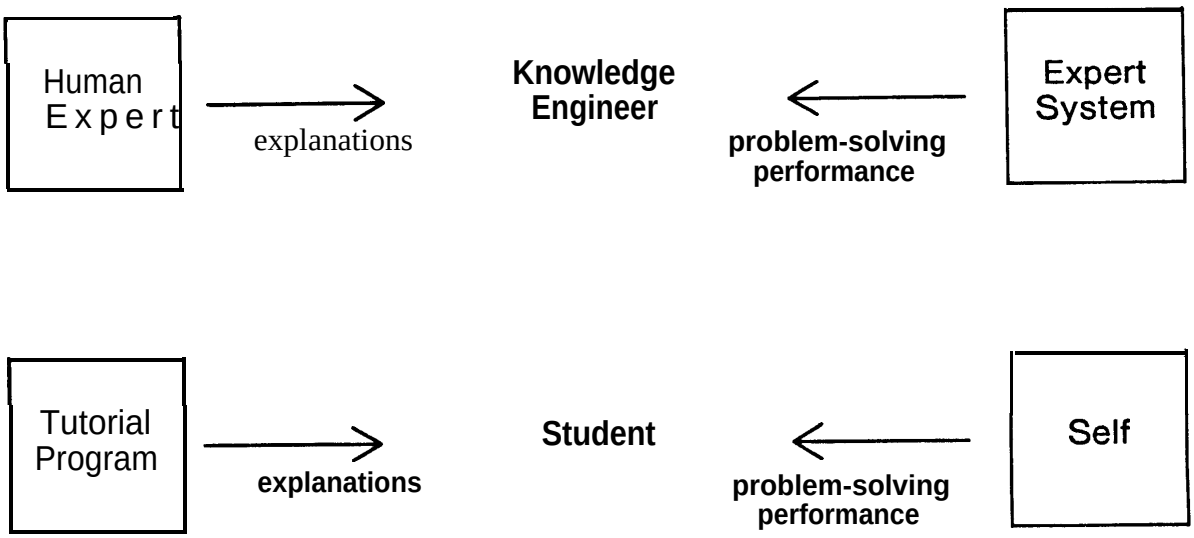

Figure l-l: Analogy of learning by a knowledge engineer and by a student: Both attend to and solicit explanations in order to improve problem-solving performance.

This study is the precursor to developing a ***learning apprentice*** knowledge acquisition program (Mitchell, et al., 1985) to assist in debugging NEOMYCIN. We then intend to develop a tutorial program that conveys this model of learning to a student while he is diagnosing medical problems. Thus, we follow the knowledge-based tutoring paradigm of first formalizing a program that can do what we will ask a student to do, specifically, to detect inadequacies in problem-solving performance and transform them into good questions for a teacher. The learning apprentice thus serves as a model for the student to study and emulate, as well as provides the tutorial program with a basis for assisting and evaluating his performance.

1.2. Origins in previous learning research

This research is strongly influenced by other attempts to teach general problem-solving methods, such as the Schoenfeld's work in mathematical problem solving (Schoenfeld, 1981) and Brown's work in algebra problem solving (Brown, 1983). However, we expect our model of learning to be better articulated and substantiated through the process of developing a simulation program. Furthermore, by working in a nonmathematical domain, we are dealing with complex knowledge structures, which we believe will better demonstrate what problem-solving knowledge transfers across domains. Current expert systems research suggests that abstract knowledge-base structures, such as relations used in the causal reasoning of diagnosis, recur in different domains (Clancey, 1988, Bennett, 1983, Chandrasekaran, 1984). These recurrent structures enable us to reuse knowledge representation languages and reasoning procedures in different expert systems, forming the basis of expert system tools called "generic shells" (Chandrasekaran, 1986), of which HERACLES, a generalization of NEOMYCIN, is an example.

The particular model of learning developed here has its basis in our previous study of expert systems (Clancey, 1986a). We describe complex problem solving in terms of a system being reasoned about (such as an electronic circuit) and a *task* by which the system is to be manipulated (e.g., diagnosis, design, control). Unlike other current research in knowledge acquisition, which uses models of knowledge organization and inference processes to direct a learning process (e.g., (Mitchell, et al., 1985, Smith, et al., 1985), we need to make the learning process itself explicit so that it can be reasoned about by the tutorial program. A computational model alone is not sufficient; its representation must be well-structured. Specifically, according to the model of learning we are developing, we must make the ***model-building process*** in performing some task on a system explicit. For example, in diagnosis problems we must make explicit the constraints that a good diagnostic model must satisfy, so we can articulate problem-solving failures to a student and relate them to the subtasks of diagnostic reasoning. None of this would necessarily be explicit in a typical expert system or learning program.

A complex domain, such as medical diagnosis, requires a complicated reasoning procedure, which provides us with many examples of how knowledge about knowledge organizatic n is used to focus reasoning and to formulate good questions when reasoning tasks fail. As our examples will make clear, problem domains like geometry theorem proving and. algebraic equation simplification are comparatively impoverished: These domains provide lit1 le ***content*** that is general, and hence provide minimal leverage for learning about other problen domains. In comparison, experience with medical diagnosis provides a substantial basis fc: learning about diagnosis in other areas, such as electronic diagnosis. It is precisely this dom .in-general

knowledge that makes a KE an efficient learner.

In particular we **want** to teach a student the organization of diagnostic knowledge and how it is used. In our research on NEOMYCIN, we have developed a model of diagnosis in which the procedure of how to do diagnosis is separated from the domain facts (Clancey and Letsinger, 1984, Clancey, 1984a, Clancey, 1986b). Thus, at each point what the program is trying to do, called a *subtask*, is translated into a question about what the program needs to know. For example, in refining a hypothesis H the program asks, "What are the subtypes of H? What could cause H?" The essential idea in the NEOMYCIN model of diagnosis is that a sequence of requests for problem data (e.g., "Does the patient have a fever?" "Has he travelled?") can be abstracted in terms of ***operators*** for manipulating a ***situation-specific model*** that describes the physical processes by which the patient's symptoms were produced (Clancey, 1984a, Clancey, 1986b). This knowledge about what the problem solver is trying to ***do*** and the structure of his experiential knowledge, as abstracted from diagnostic practice and formalized in NEOMYCIN, is the essential ***metacognitive knowledge*** we seek to exploit. The sequence of knowledge engineering examples in Section 3 builds up to this model of diagnosis, illustrating how knowledge about diagnostic knowledge and its use, a form of metacognition, provides a basis for active learning in the KE.

1.3. Limitations in the analogy between a knowledge engineer and a student

It is important to realize that there are significant differences between a typical student's task and a KE's task. A KE is constructing a computer program, she is not learning how to solve a problem independently. Specifically, a KE does not have to remember everything she puts into a program. In working with an expert, she gains very little experience in proficiently integrating everything she has been told. Rather, she tends to view facts and problem-solving procedures in isolation, as they apply in specific cases. The implications of this difference are not immediately clear, but must be attended to later in evaluating the model of learning we develop. If we are lucky, the only difference will be the practice effect of solving problems from memory, and the program and student who follows the KE's model of directed learning will exhibit no difference in content of what is learned or in problem-solving ability.

We also know that students generally have a substantial background of factual knowledge about a domain. For example, medical students have 2 years of general learning about disease processes before they focus on practical, clinical problem solving. In contrast the KE has comparatively less specific knowledge about physiological processes, and the computer program has essentially none. Thus, another weakness in our analogy and potential barrier in our attempt to automate the KE's learning process is the absence of general knowledge about physical processes that a student, and to a lesser extent the KE, can draw on. If we are lucky

here, this will impoverish the model of learning we develop, but not detract from the general form of the model (e.g., its basis in problem-solving failure) or the specific results concerning the learning of routine problem-solving knowledge (e.g., the emphasis on well-defined relations that carry across domains), which we focus on.

On the other hand, the essence of our approach is that a KE learns by filling in a schema, his knowledge representation, which is rarely known explicitly by a medical student. This is the KE's strong suit, which we seek to exploit and articulate in a simulation model of learning.

Another difference between the typical student and KE is that the theory of problem solving developed by the KE is the result of experience in different domains. It is not immediately clear whether we can make the KE's problem-solving abstractions understandable within the context of a limited set of problems in just one domain. If we are lucky, the value of these abstractions will be apparent; and possibly the process of formalizing them from our experience in constructing the first expert systems was just a one-time difficulty that other people will not need to repeat.

We also start with the hypothesis that a student does not need to know a diagnostic strategy explicitly to become proficient. Indeed, medical expertise has certainly advanced and teachers have been effective without being able to articulate the diagnostic process to the extent that it is formalized in NEOMYCIN. We argue that knowledge of the diagnostic process is useful not for routine problem solving, but for recovering from failures, the essence of the model of learning we describe. A corollary is that we are teaching a method of learning that medical students typically do not use or at least do not use systematically.

One advantage of studying KEs, versus typical medical students, is that a KE is always working with a formal representation (the knowledge base). We can directly observe how she manipulates this representation and relate her behavior to the current state of her computer model. The formal representation provides a language for systematic description of what the KE is doing, which we view as a learning process. In developing a tutoring system, we will investigate the benefits of giving a student similar written notations to use in recording his understanding and solution to particular problems.

To recapitulate, we argue that the KE's interview ability and program development process follows from her representation of how knowledge is organized and used in solving routine problems. This metaknowledge is coupled with a procedure for detecting knowledge gaps that critiques incomplete problem solutions. From this basis a problem solver can direct her own learning by formulating good questions-those that are directed at what she needs to know.

2. The map metaphor: Representations, models, and problem solving

Before considering complex examples of metacognitive knowledge and how it is used in learning, it is useful to develop our intuitions by considering a familiar example. In particular a road map nicely illustrates the nature of representation and inference procedures. The familiar nature of a map is extremely useful for revealing the way in which an expert system's knowledge representation is a model of the world. This is especially important in our argument because the idea that an expert system knowledge base contains a model of the world is mostly ignored by AI researchers, to the extent that the term "model" is generally only used to refer to simulations of processes. A few of these ideas are explored here, others are developed in (Clancey, 1986a).

Figure 2-l is a portion of a road map. A map, like any ***representation,*** makes a commitment about the existence of particular classes of objects and relations in the world. That is, the map's notation-the particular symbols used in the map-allow certain statements to be made about the world and do not provide for others. For example, the map in the figure allows different kinds of buildings, roads, and facts about how they are related to be represented. We see that the faculty club is on Laguna Drive and that Memorial Hall is adjacent to the Graduate School of Business. However, this particular notation does not provide a means of indicating the kinds of soils found on this land or possible ore-bearing deposits. This is a suitable representation for moving a car around campus, but not for drilling for oil. Thus, a map categorizes the world in a certain way; only certain distinctions can be expressed.

More formally, a map notation is defined by a set of terms (objects or spaces) and relations among them (notably connectivity, size, and distance). These terms and relations constitute a language for articulating propositions (believed statements) about the world, called the ***representation language.*** A specific map is a ***model*** of the world. Crucially, to this point, everything we have said about maps is true about expert system knowledge bases.

The ***semantics*** of a map concerns its meaning, that is, how it relates to the world. For example, the closed solid figures stand for or represent buildings in the world. Considerable philosophic debate concerns whether it is possible to represent the semantics of a map (e.g., what is a building?). While not irrelevant to this paper, the topic goes beyond the scope of what we can consider.

Maps, like knowledge representations, are not just artwork that we carry around in our pockets. Instead, there is always some accompanying procedure for using the map to solve practical problems. For example, a road map is associated with procedures for planning a trip from one location to another. We call such interpretation procedures ***inference procedures.***

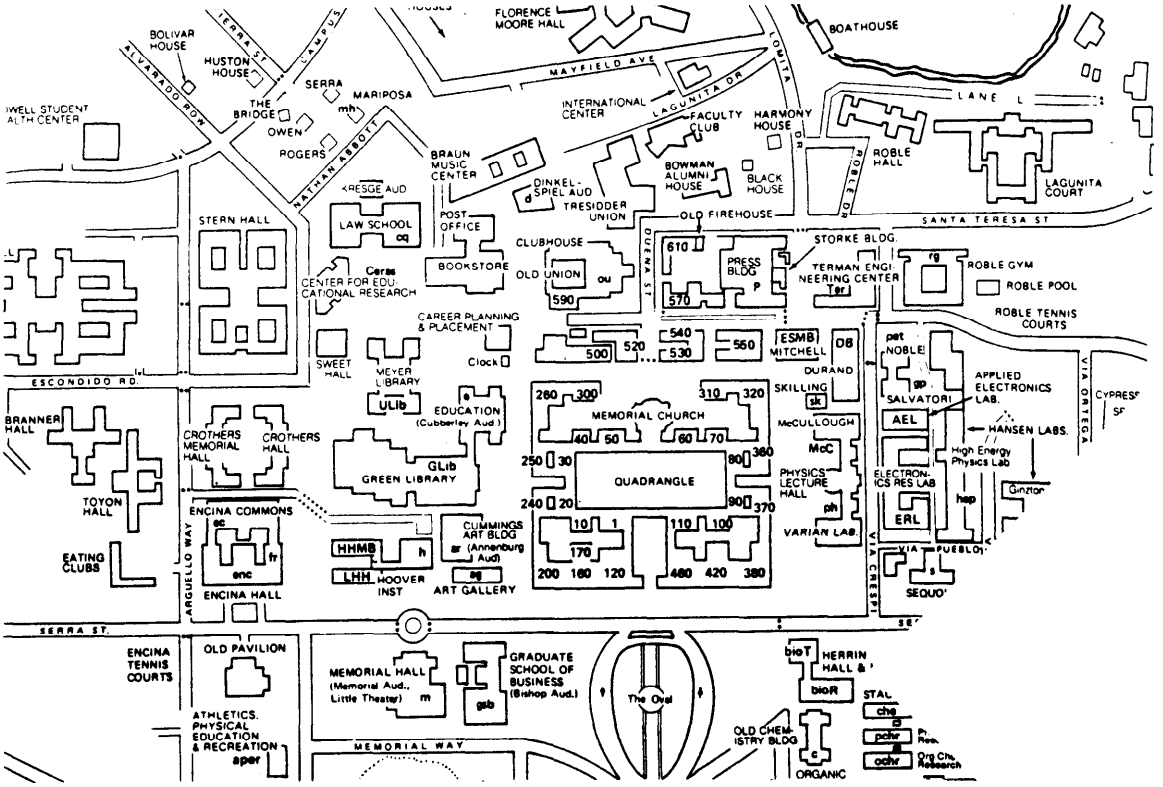

Figure 2-1: Portion of a map

This procedure (or program) makes extensive use of the language of the map. The inference procedure indicates how to use the map to solve a practical problem.

Sometimes we describe an inference procedure in terms of how it controls search (thus, the synonym, "control knowledge'*). Of all the possible questions we might ask about the world (data we might find useful for solving a problem), and out of all the possible inferences we might draw from our map, which ones should we consider first? And which after that? An inference procedure ***orders*** data-gathering and assertions about the world. (Note that searching the knowledge representation **itself—e.g.,** determining whether there is a fire station on campus-is a different issue, pertaining to ***encoding*** of the knowledge, that is, how it is stored and retrieved.)

For example, consider using the map in Figure 2-l for moving your car from the Old Pavilion to the Faculty Club. Which parts of the map do you look at first? What controls those observations? What partial plan do you first construct? Why do you form those pieces first? This procedure, which tends to recur when we solve similar problems, is the inference procedure. In this example it is a planning procedure. Different kinds of problems have different characteristic inference procedures. Perhaps the most well-known is the procedure for diagnosis, also called a diagnostic strategy (Clancey, 1984a).

Inference procedures constitute a form of experiential knowledge. To develop this point,

consider first that the representation language (to be contrasted with a particular map or **knowledge** base) is associated with operators for making inferences (often called ***inference rules).*** **For** example, if we need to determine the distance between two sites, we use the distance and adjacency information implicit in the map. Thus we make a statement about the world by piecing together more primitive relations, as well as by extracting implicit facts. For any given map in a particular language, used for a particular kind of problem, a given set of operators will be useful. For example, in using roadway maps, we repetitively need to determine distances, the nearest object of type X to a given object, the shortest path between two objects, and a few others. The number of operators of this form is not necessarily large.

An often referred to distinction is that some inference operators are ***heuristic,*** as opposed to ***definitional.*** For example, "If classes are changing, do not attempt to cross Escondido Road by car" is a heuristic that could affect the solution of the parking problem. Characteristically, such a heuristic does not follow from the meaning of the map (the definitions of the symbols), but involves experiential information about cause-effect relationships, involving other objects in the world that are not represented in the map.

Crucially, the way in which inference operators are chained together in making inferences (to construct a plan) tends to recur. Thus, in moving from one place to another, we generally tend to consider the shortest path, depending on our mode of transportation, and will build a path from one end to the other, using methods like finding the largest artery between the two locations and anchoring the path by other places we wish to see along the way. These methods are to be contrasted with a comparatively unintelligent process of constructing multiple paths and making useless inferences, such as considering the distance between arbitrary points unrelated to the movement under consideration. Similarly, in planning a walk, different questions will be asked about the world. For example, we will be less concerned with unmarked roadblocks and traffic congestion, and perhaps more interested in where people tend to sit or congregate, so we can have a more pleasant passeggiata.

In summary, the inference procedure indicates heuristically how to extract useful facts in order to solve the problem at hand. A domain-general inference process is focused on what we are trying to do and most specifically on the constraints we seek to satisfy. Other considerations that we cannot consider in detail here concern the nature of the situation-specific model (e.g., do we write down our problem solution in some notation?) and the nature of implicit facts in the notation. Finally, as a representation of the world constructed for a particular purpose, the map is a selective model of the world. It simplifies the world in a particular way so that problem solving can proceed efficiently. Representations are therefore biased by the degree to which they are specialized for particular tasks.

Briefly, to make the analogy with knowledge representations more explicit, **NEOMYCIN's** knowledge representation language consists of a set of terms (e.g., finding, hypothesis, test) and relations among them (subtype, cause, abnormal finding). The language is associated with an inference procedure that solves diagnostic problems by the general method we call ***heuristic classification*** (Clancey, 1985). The language is biased (specialized for diagnosis) and is more familiar to physicians than engineers. In a rough sense, knowledge-base networks of concepts and relations organized into hierarchies and transition graphs are analogous to the objects and lines of a road map. A knowledge base representation language makes distinctions that are useful for solving particular classes of problems by particular inference procedures.

Here we are most concerned with the value of a representation language for learning practical problem solving knowledge, not arbitrary facts about the world. Furthermore, we are concerned with how a ***given*** representation is useful, not with the process of modifying a representation. This is an ***assimilative model of learning,*** assuming no representation change (Norman, 1982).

The first key observation is that a given representation literally provides a language for asking questions about the world. Consider using the map language to learn about a new area. In learning that a building is in a certain location, you might ask what roads are closest to it, where is the nearest parking lot, and what buildings are adjacent to it. The language critically influences what we know about the world. (The much-debated philosophic point, generally associated with Whorf (Whorf, 1956), about how language shapes knowledge and experience shapes language, goes beyond the scope of this paper.)

The second observation is that construction of a specific model is intimately tied to the goal of solving a problem. That is, we realize gaps in our knowledge of the world when we are applying an inference procedure and find that the required facts are missing; there are gaps in our map of the world. For example, a medical student might conclude that a patient has a chronic-meningitis infection and know that this is not specific enough for prescribing therapy. His inference procedure indicates that he has not solved the problem, and he should now apply the operator for refining a disease hypothesis (to make it more specific). Yet, at this point he may realize that he does not recall the subtypes or causes of chronic meningitis. He realizes that there is a critical gap in his knowledge, an ***impasse*** (following the terminology of (Brown and VanLehn, 1980), an analogy with subtraction we develop later). This example illustrates that, by definition, useful distinctions-what you need to know about the world-are not based on gaining know edge for its own sake, but arise in solving a particular problem and are directly related tc the inference procedure being applied.

To recapitulate, the model of learning developed in this paper concerns how knowledge about

a representation language and inference procedure enable the learner to articulate what knowledge he needs to know and hence to formulate a specific question for a teacher (e.g., "What are the subtypes of chronic meningitis?"). We will return to a more specific description of this process after considering how knowledge about representations and inference procedures is used by KEs.

3. Learning heuristics used by knowledge engineers

This section briefly surveys a variety of heuristics used by KEs for learning about a new domain and improving a knowledge base. This learning process is typically called ***knowledge acquisition.*** The sequence of examples presented here illustrates how the nature of active learning changes with the available representation. In particular we observe the progression from the terms of the rule-based representation language of MYCIN to the heuristic classification language of NEOMYCIN. The final example, in which diagnosis is described as a model-construction process, illustrates the advantages of abstracting the inference procedure so that the representation or map of the world is stated declaratively, as facts, that are separated from the procedure for using them. This separation makes explicit the representation language as an abstraction that is available for use in new problem domains.

3.1. Guidon orientation tutoring

In approaching a new domain, a ICE developing an EMYCIN system might ask the following sequence of questions over the course of several interviews with the expert, and continue to pursue them while watching the expert solve particular cases:

- What is the goal rule?
- . What is the main subgoal structure?
- What do premises of the rules look like? Are there patterns?
- What are the important input data? What kinds of judgments are required of the person supplying data to the expert system?
- What are some typical outcomes for the major subgoals, and what are some typical rules to conclude about these outcomes?

When introducing a new problem to a student or discussing a subproblem, GUIDON (Clancey, 1987) provides orientation by effectively inverting the preceding questions, presenting the same material an experienced KE familiar with the EMYCIN rule language actively seeks from an expert (his teacher). In addition to showing the goal rule, the program indicates the general subgoal structure for the case at hand (indicating only subgoals used by or concluded by a large number of rules) (Figure 3-l). Each new major topic is introduced by a definition, outline of major subgoals (called a ***rule model),*** and typical values (Figure 3-2). This information helps

a student understand how the diagnostic problem has been formalized into objects and relations.

3.2. Interviewing an expert: Structure, Strategy, and Support

In the process of studying MYCIN's knowledge base to determine how it might be improved to be more effective as a basis for teaching, we developed a framework based on an expert's explanations of diagnostic reasoning. Explanations are analyzed according to ***knowledge roles-how*** knowledge is used in relation to other knowledge (Clancey, 1983a):

1. The ***heuristic rule,*** a relation between data and diagnoses or therapies
2. ***Structure,*** subsumption relations among data, diagnoses, and therapies
3. ***Strategy,*** the procedure for applying rules
4. ***Support,*** the justification for rules

These categories provide guidance for listening to and directing a teacher. They provide a means for understanding how a teacher's statements are related, so the student can organize what he is hearing and focus the teacher to fill in other connections he needs to know. In particular, it is useful to cut short detailed support justifications and instead focus the teacher on structural overviews and their strategic motivations.

For example, when the KE asks a physician who is solving a diagnostic problem, "Why did you request that information?" she classifies his answer into one of these categories. If he tells her, "Well I'm not going to prescribe tetracycline because the patient's age is less than seven," she is being told what assertions were made from given information (i.e., a heuristic rule: If the age is less than seven, tetracycline is contraindicated). If she asks him why, and is given an explanation having to do with chelation, then she is being given the justification for the assertion (i.e., support, a chelation process causes teeth discoloration). If the expert says, "This is just one of the ***contraindications*** I'm going to consider," then he's telling the KE about the organization of his knowledge, the categories he uses for focusing (i.e., structure, undesirable physical changes are contraindications for drug therapy). Next, if he tells her ***when*** he considers contraindications and ***how*** he considers each type, then she is being told the inference procedure (i.e., strategy, after hypothesizing a drug therapy, attempt to rule it out by considering contraindications).

The strategy-structure-support categories can be restated as more active, specific heuristics for directing a teacher. We collected such heuristics by analyzing protocols of interactions between a KE and a physician-teacher. The KE's questions are classified to reveal her methods for actively critiquing, testing, and refining her understanding, while watching the teacher solve a problem.

Sketch of the tree of subgoals for determining the therapeutic regimen of J.Smith:

7a. The therapeutic regimen of J.Smith
- 7b. The organisms (other than those seen on cultures or smears) which might be causing the infection
- 7c. Whether J.Smith has a head injury defect
- 7d. The infection which requires therapy
- 7e. The type of the infection
- 7f. The diagnoses of J.Smith
- 7g. Whether organisms were seen on the stain of the pending csf culture (CULTURE-1)
- 7h. Whether the organisms isolated from the pending csf culture (CULTURE-1) should be considered for therapy
- 7i. The organisms that will be considered to be causing the meningitis (INFECTION-1) for the purposes of therapy selection

You can produce a full tree of subgoals for any topic by the SUBGOALS option.

Figure 3-1: Guidon's presentation of subgoals for a given case

```
You should now proceed to ask questions that will enable you to make an
hypothesis about the organisms that might be causing the infection.

A pathogenic organism which was not seen on a culture or smear but
which might be causing an infection in the patient and therefore requires
antimicrobial therapy is an organism to "cover for."


  CULTURE-1 is a pending culture.  In this context, when we are
considering the organisms that might be causing the infection, we
generally find it useful to consider:

        8a.  the infection which requires therapy
        8b.  the type of the infection
        8c.  whether organisms were seen on the stain of the culture
        8d.  whether a smear of the culture was examined

There are 93 rules used by MYCIN to conclude about this topic.
Altogether there are 29 factors considered in these rules.

Some sample values for the organisms that might be causing the infection
are: proteus-non-mirabilis, streptococcus-group-b, proteus-mirabilis and
neisseria-meningitidis, etc.
```

Figure 3-2: Guidon's orientation for a new topic

- **Strategy**
  - Ask about ordering of data requests.
  - Determine specificity of reason for data request (a general question or directed at a particular problem solution?).
  - Track hypotheses and detect when focus changes.
- **Structure**
  - Ask for typical and atypical problem examples (cases).
  - Listen for exceptions immediately after a general rule is stated.
  - Detect synonyms.
  - Summarize your understanding.
  - Beware of implicit assumptions (what is the expert inferring from context?).
- **Support**
  Ask for cases with inconsistent findings; listen for rationalizations of these in terms of possible misconceptions or misunderstanding (e.g., "I might be wrong about the prevalence of this symptom")

As always, the distinctions drawn here presuppose a model of useful knowledge relations (e.g., "inconsistent findings"). In particular, many of the preceding heuristics might be justified by a model of knowledge organization, that is, a description and generative explanation for the recurrence that occurs in the expert's memory associations. (Another philosophic consideration beyond the scope of this paper argues that such associations are not ***prestored*** but rather

generated at the time of knowledge articulation, as suggested by (Winograd and Flores, 1985) and reflect general properties of narratives, as suggested by (Bruner, 1986).) The knowledge organization and inference procedure implicit in the preceding heuristics are partially formalized in the heuristic classification model of diagnosis.

3.3. Knowledge acquisition for heuristic classification

The rule and goal language of EMYCIN has been specialized in the heuristic classification model of problem solving (Clancey, 1985). Distinctions are made between findings, hypotheses, and classifications of these (concerning type, causality, abnormality, location, etc.). Distinctions are made between domain rules, definitional rules, and causal rules. In addition, goals and rules that represent the inference procedure are called tasks and metarules. This language structures the knowledge base, providing a more specific language for articulating the cause of problem-solving failures. In particular, the following set of ordered heuristics was developed from the experience of constructing several knowledge bases in the Heracles language (Clancey, 1984b):

1. ***Problem selection:*** Selecting a domain, task, and scope for the expert system.

   - Look for problem types that can be solved by classification: Are the solutions enumerable and stereotypic (configurations, plans, diagnoses, etc.)?
   - Decompose problems into sequences of classification problems. Treat them separately, but work backward from the final problem. For exam ple, needs/requirements analysis may be solved by classification, with a solution heuristically related to ultimate solutions (products, services, etc.): work backward from these solutions. Another common example: Consider what kinds of repair are possible before analyzing the associated diagnosis problem.
   - Early on, define the problem in terms of input and output and the kinds of relations. Try to distinguish between substances and processes. What is observable? Does causality play a role? For diagnosis, is there a disorder or abnormal state network? Is there a hierarchy of disorder processes (what can go wrong)?

2. ***Knowledge-level analysis:*** A structured way for identifying terms and relations.

   - List all possible solutions the program may output; organize into classes and hierarchies if appropriate. Be clear about what the solutions are: plans, processes, configurations, and so on. **A** confusion at this point may mean that there are separable problems. Be clear about ***types,*** that is, do not mix different kinds of things (e.g., descriptions of diseases with descriptions of people). All solutions should at a high level belong to a single class.
   - List classes of data that will be input to the program (no need to be exhaustive at this point, unless the list is under a few dozen items); will any data be numeric? Organize into classes and hierarchies to the extent possible.
   - Identify relations among the data: generalizations, definitions, and qualitative abstraction. Exact attention to relations is difficult but essential to be sure the problem is adequately decomposed. For example, ♦ care to distinguish

an abstraction of data according to definition from ***an abstraction of the solution*** that is matched by direct identification of some features. (For example, "white blood count less than 2500" is the ***definition*** of 'eukopenia [a data abstraction]; "gram-negative rod" ***matches the features*** of E.Coli [a diagnostic solution].) A common problem is that the expert will leave out qualitative abstractions, stating associations in terms of numeric data, or vice versa, not indicating until later that data are actually numeric.

- Establish the heuristics that link data to solutions ***after*** establishing the network of solutions. To avoid identifying a solution as a datum, be aware that some rules may relate solutions to one another nonhierarchically (in diagnosis this is called a ***complication).***

- Treat the inference process separately. It is essential to model the expert's inference structure (terms and relations), but not as important to model the inference process he uses. For example, a program may use top-down refinement within a hierarchy of solutions, while the expert may use a more opportunistic, hypothesis-formation approach. Modeling inference ordering is much more difficult, and is in general not necessary for efficiency in expert programs of the size constructed today.

3. ***Implementation:*** It is advantageous to use a programming language that allows relations to be made explicit, especially hierarchies. Top-down refinement can be easily encoded by ordering rule premise clauses, but this approach leads to redundant, more complex rules, with a loss of explanation capability. Better engineering suggests separating the inference and process structure (Clancey, 1983a).

4. ***Knowledge-base refinement:*** The classification model suggests selecting cases that will test the program's ability to discriminate among solutions, consistent with the usual approach of improving the knowledge structure by testing the program on a variety of problems. One might begin with classic cases corresponding to each solution, then systematically pick problems with similar input, but different solutions.

Note that this is a framework for systematically describing knowledge, not for eliciting it. For example, the order of knowledge-level analysis given here (basically, a bottom-up, output-to-input approach) may be a useful organization for the learner, but the expert may find it difficult to directly describe what he knows in this way. It may be preferable to present problems to the expert and quiz him about what he is doing.

The distinctions given here can be viewed as specializations of the explanation categories (Section 3.2), in which structure and strategy are the relations and tasks, respectively, of the heuristic classification inference procedure. This model of experiential knowledge is very general. It is a much more useful description of what we need to teach a student than is provided by EMYCIN's knowledge representation language, in which we can only say that knowledge consists of rules, goals, input data, values, and so on, and specific instances of these.

3.4. Constructing a causal state network

The heuristic classification model can be specialized for diagnostic tasks. For such problems the findings are symptoms, and the hypotheses are abnormal processes or states in a system being diagnosed. Further relations characterizing findings include "red-flag finding,*' "normal value," and "causal prerequisite finding." The inference process is specialized from the language of inference graphs to the language of diagnosis; rather than saying "backward chaining," we say "reasoning from symptoms to abnormal states to abnormal processes." Operators for traversing the network of hypotheses (a kind of map) include refine, test, rule-out, group, and discriminate.

Active-learning heuristics using this knowledge representation and inference procedure language include the following (using examples from CASTER, a sandcasting diagnosis program built in the HERACLES shell (Thompson and Clancey, 1986)):

- ***Identify the fundamental terms and relations in the domain before writing rules.*** For example, sandcasting involves substances like sand, water, gases, and metals; processes like melting metal, designing a pattern, building a mold, and pouring metal; forces like gravity and gas pressures. After these domain terms are introduced, the KE is ready to learn about causality between processes, and refine her knowledge of disorder types.

- ***Ask about categories of substances and processes.*** Often the heuristic (causal) relations between data and solutions are stated as generalizations relating these categories (e.g., different kinds of bubble defects are caused by gas, which has different sources).

- ***Describe abnormal events in terms of temporal phases.*** For example, in manufacturing abnormal events can be categorized according to the processes that cause them. For example, the problem of inadequate feeding of iron in sandcasting occurs during the freezing process.

- ***Identify abnormal properties of substances, then seek causes.*** For example, metal contamination is a serious problem in casting. After identifying a number of possible contaminants, such as aluminum, silicon, and phosphorus, it is appropriate to consider how each type of contamination might occur and evidence for specific types (their manifestations). Patterns will enable learning specific relations more easily by analogy if the categories are established first.

- ***Identify possible malfunctions, determine the corrections for those problems, and then causally reformulate the relationship.*** For example, for the problem of "feed shut off" we ask what possible changes could remedy the situation. These include making the gates bigger, the neck of the riser bigger, fillets larger, the metal hotter, and so on. These repairs are then reformulated in terms of how they will change the manufacturing process, and hence how the previous processes caused defects.

- ***Establish causality between findings and malfunctions, then expand intermediate relations.*** By making relations explicit (as measured by need in actual cases), it is possible to discover generalizations that collapse many specific heuristic associations into a common mechanism.

In summary, the preceding heuristics can be viewed as things to do to improve a causal model-ways of usefully critiquing a partial model of disorders. Reflecting back on the examples given of **GUIDON's** orientation, it should be obvious that these heuristics can be turned around for presenting information to a student or probing his understanding. More in keeping with the active model of learning we are developing here, these heuristics could be applied by a student himself to articulate to a teacher what he needs to know, which is precisely what the **KE** does.

3.5. Graphic representations

Although it is well-known that a good representation can greatly affect problem-solving efficiency and even the possibility of solving a problem, the most common examples involve puzzles such as the problem of tiling a checkerboard (Jackson, 1974), rather than the **kinds** of representations used in expert systems. Here we illustrate how graphics can be used to make salient the patterns (relations) among objects, prodding a student to formulate a ***generative model,*** which explains why objects fall together in the same group. These graphics are contrasted with the "modular," linear form of individual rules which provide no basis for realizing the presence of patterns, let alone expressing them explicitly. This point is clearly illustrated by the shift from **MYCIN's** goal-rule language to classification relations in **NEOMYCIN,** which articulates and explains subgoal patterns in **MYCIN's** rules (Clancey, 1983b).

3.5.1. Viewing process similarities by overlapping line graphs

A striking example of the inadequacy of a verbal representation for revealing knowledge relations is a typical tabular rule in **MYCIN,** "the cerebrospinal fluid (CSF) protein rule" (Figure 3-3). Faced with the difficulty of understanding this rule, we graph it, expressing the same knowledge by a pictorial representation (Figure 3-4). It is now apparent that some disease processes are similar. That is, the representation indicates no distinction between them over

certain CSF protein ranges, within a certain tolerance of change. The physician teacher stated the principles this way, "If the protein value is low, I think of an acute process; if it is high, I think of a severe or long term process." (Bacterial meningitis is a severe, acute [short term] problem, while fungal and TB meningitis are problems of long [chronic] duration.) Generalizing this further, he describes the common mechanism, "An infection in the meninges stimulates protein production."

This example illustrates again how a given representation will naturally lead us to generate certain kinds of questions about the world. In a graph of this form, these questions include:

. What accounts for maximums? (Is it unusual for the CSF protein to be greater than 300? Why is belief concerning bacterial meningitis never greater than 0.4?)

- What accounts for zeroes? (Why do TB and fungal cross from negative to positive belief at 41 mg/ml?)

- What accounts for patterns? (What do bacterial, fungal, and TB have in common that is not true about viral?)

The use of graphs for scientific theory formation is well-known. Such representations structure facts about the world, revealing patterns that we can then seek to understand. A representation provides a means for articulating our experience, structuring what we believe to be true about the world. By biasing the patterns that can be expressed, different representations reveal different similarities, leading us to ask different questions about the world (why certain patterns exist). In contrast with the well-known scientific techniques, the model of learning we are developing is couched in terms of a simulation model of problem solving. That is, our aim is not to merely seek and explain patterns, but to solve practical engineering problems.

One effect of detecting new patterns is that we add new relations to our language to express them, changing our representation. For example, useful abstractions for understanding the CSF protein rule include "high protein" and "chronic process." The original patterns (the graph) can be generated from this more abstract representation, which is what happens when NEOMYCIN replicates patterns in MYCIN's rules when it refines hypotheses, generalizes data requests, checks to see if a test is done before requesting specific results, and so on. (The process of changing a representation by abstracting an inference procedure is described in (Clancey, 1986b).)

RULE500

If: 1) The infection which requires therapy is meningitis,
2) A lumbar puncture has been performed on the patient, and
3) The CSF protein is known

Then: The type of the infection is as follows:
If the CSF protein is:
a) less than 41 then: not bacterial (.5), viral (.7),
not fungal (.6), not tb (.5);
b) between 41 and 100 then: bacterial (.1), viral (.4), fungal (.1):
c) between 100 and 200 then: bacterial (.3), fungal (.3), tb (.3);
d) between 200 and 300 then: bacterial (.4), not viral (.5),
fungal (.4), tb (.4);
e) greater or equal to 300 then: bacterial (.4), not viral (.6),
fungal (.4), tb (.4);

Figure 3-3: The CSF Glucose rule, illustrating **EMYCIN** tabular rule format

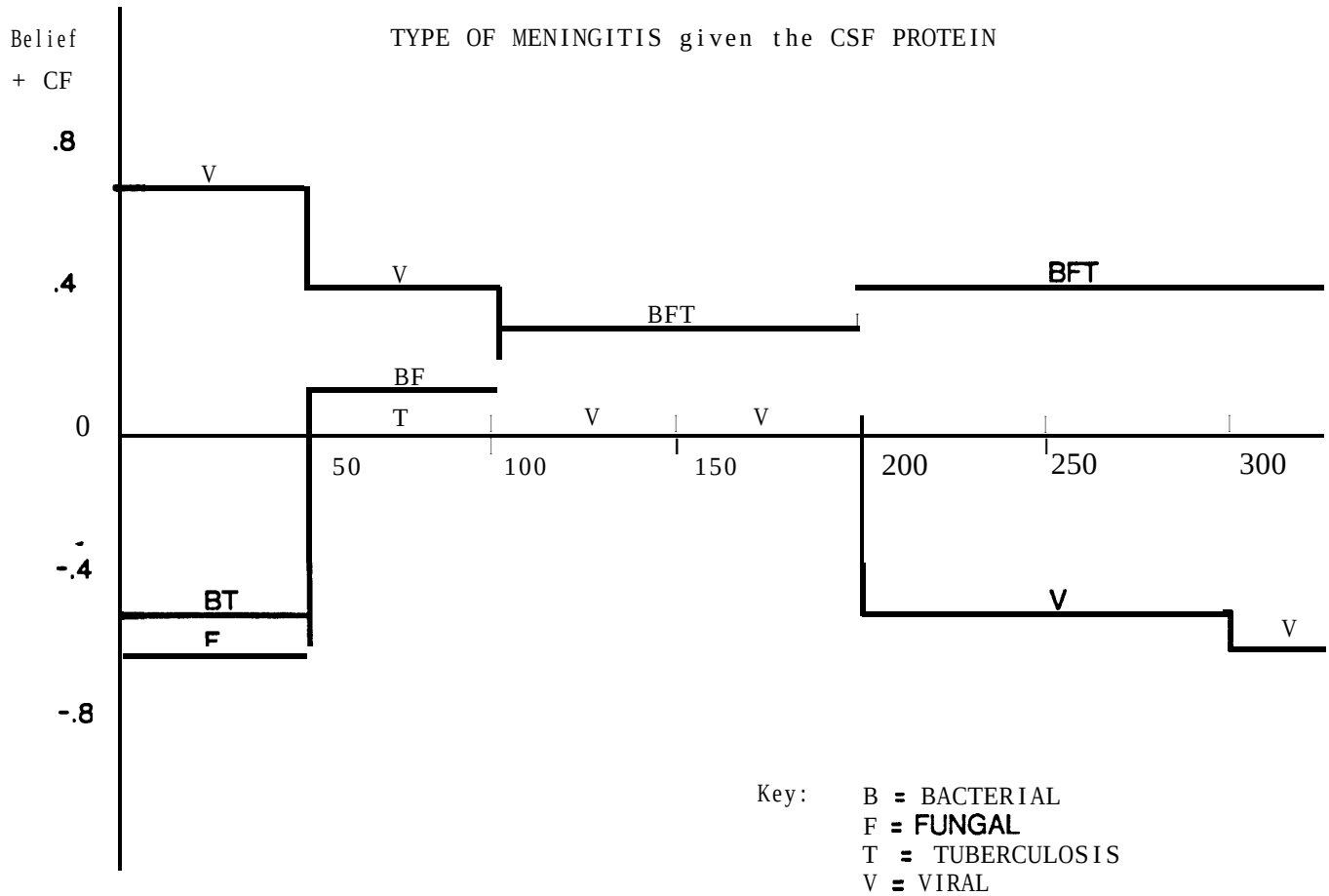

Figure 3-4: A graph used to reveal similarities among diseases

3.5.2. Viewing a diagnosis as an explanation graph

The second striking example of the value of graphics over linear rules for expressing knowledge is the use of a graph to reveal the evidence relations between- findings and hypotheses in NEOMYCIN (Figure 3-5). By this perspective, a diagnosis is not the name of a disease, but ***an argument*** that causally relates the manifestations that need to be explained (because they are abnormal) to the processes that brought them about. Our knowledge representation is thus further refined to classify diseases as kinds of ***processes,*** and to view findings as ***events*** in the world. The inference procedure "copies over" these general concepts and relations of the domain knowledge base to construct a ***case-specific model*** (Patil, et al., 1981). This network links manifestations and diseases, constituting a model of a particular sequence of events in the world (also called a situation-specific model).

Diagnostic operators (HERACLES subtasks) examine and modify the ***differential*** (the set of most-specific diseases under consideration), linking and refining state and process descriptions to construct a situation-specific model. **A** causal explanation thus has the structure of a geometry proof: It must account for all of the findings and must be coherent and consistent. The situation-specific model must be a connected graph with one process at the root (assuming a single fault). These are the constraints the diagnosis must satisfy, the ***form of a solution.***

For the purpose of teaching, this graph could be the an effective way to reify the process of diagnosis. For several years, inspired by Brown's emphasis on "process versus product" (possibly derived from Dewey (Dewey, 1964)), we have been searching for some written notation that we could use, something analogous to algebra, to make visible what the operators of diagnosis (HERACLES subtasks) are doing. The analogy with geometry turns out to be stronger than the analogy with algebra because each inference itself relies on a proof, analogous to the causal arguments behind each link of the situation-specific model. In algebra the inference rules are axioms.

Giving this window to the expert-teacher, the KE directs him to explain his diagnosis by posting his hypotheses and linking them to the known findings. Each step along the way, there are visible problems to be solved that the KE and expert-teacher can use to focus their respective questions and explanations. The use of this perspective for constructing a computational model of active learning is developed next.

4. Model of the active learner

To this point, the most specific representation we have presented is the substance/process language developed while improving the CASTER knowledge base. However, the inference procedure of HERACLES, the most developed available to us, does not use this language, but rather the more general language of heuristic classification. For example, rather than referring

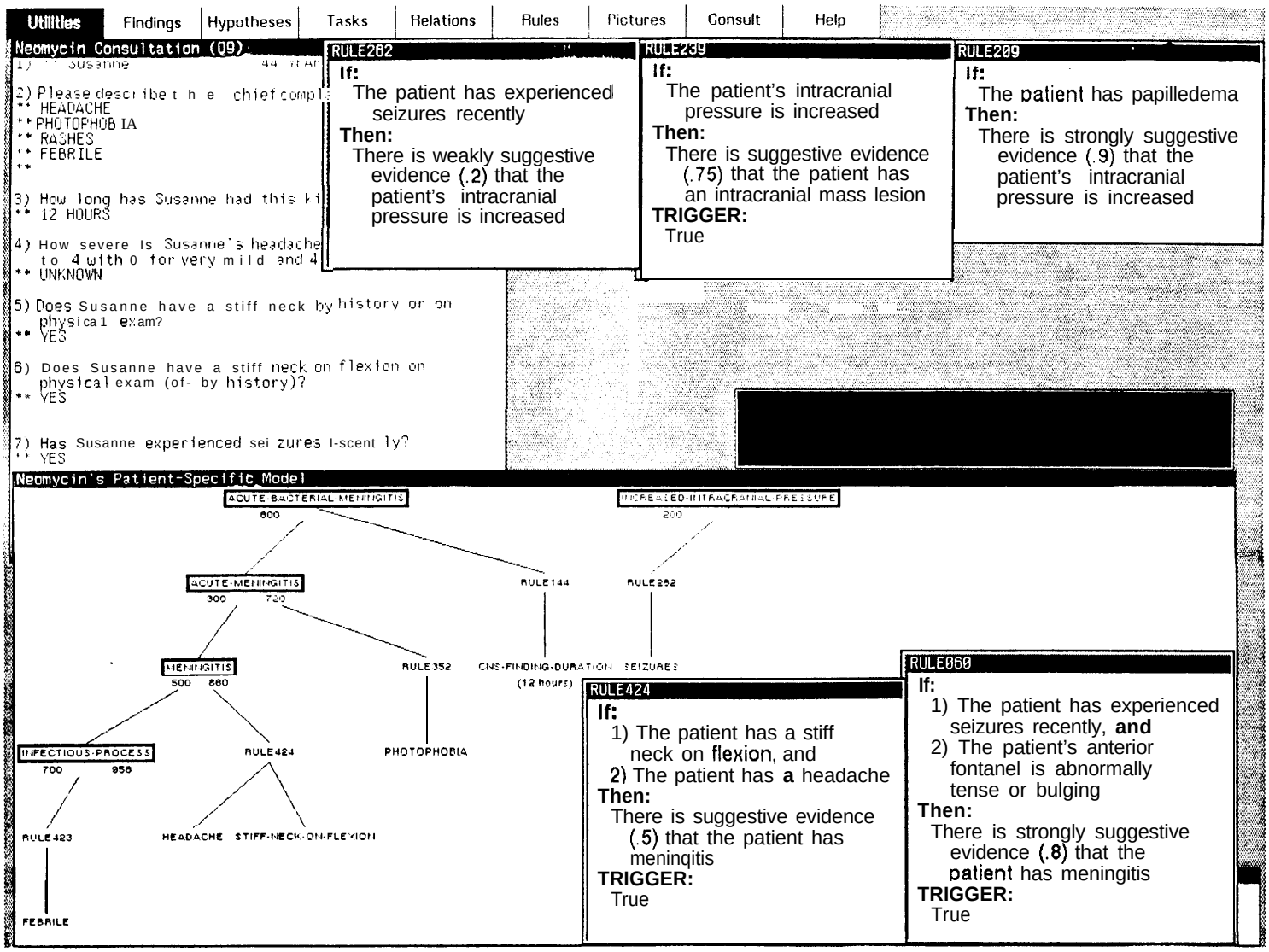

Figure 3-S: A diagnosis expressed as a patient-specific **model** in the form of a graph.

In this case, data-directed reasoning from the chief complaints (questions 2-6) led to the application of rule424, putting meningitis into the putient-specific model (called triggering). Meningitis is supported by evidence for its more general category, infectious-process. The meningitis explanation is further refined by ucute-meningitis and acute-bacterial-meningitis, each explaining other specific complaints. Further support for meningitis was sought via rule060, leading the program to usk question 7. Data-directed reasoning, through rule262, then put increased-intracranial-pressure in the putient-specific model. The program attempts to reason forward to explain this hypothesis, but before applying rule239, it seeks further support for increased-intracranial-pressure, through rule209. This leads to a question about papilledema, which is not shown.

to "substances" and "processes," it uses the terms "findings" and "hypotheses." The model of learning we develop from HERACLES therefore will be somewhat more general and less powerful (from the sense of focusing the learner's articulation of missing knowledge) than our more informal understanding of the knowledge and reasoning process of diagnosis.

Furthermore, even within the more general language of HERACLES, the view of inference operators as model-construction operators, as developed in the previous section, is not captured. Without this perspective our model of learning is impoverished, since it does not make explicit the constraints that problem solving seeks to satisfy. This is reflected in the fact that NEOMYCIN does not detect that it has not explained findings or offer any evaluation about the adequacy of its solutions (beyond the strength of the "evidence" for hypotheses). Therefore, although we include in the model of learning developed here the view of problem solving as a model construction and application process, the reader should keep in mind that implementing this will require modifications to the HERACLES shell, as detailed following.

In the discussion that follows, notice that "the student" and "NEOMYCIN" are interchangeable.

4.1. Formalizing the learning process as a knowledge acquisition program

We want to teach the student domain knowledge that will enable him to solve problems by heuristic classification, a specific knowledge representation and inference procedure in terms of which all knowledge will be expressed. For example in the language of HERACLES, the student will learn classifications of findings and heuristics to relate them to classifications of solutions. He will learn to recognize and discriminate these prototypes. Using knowledge of the heuristic classification representation and inference procedure, the student will explain his failure to solve problems and direct a teacher to supply him with the facts about the world that he needs to know. A basic assumption is that learning will be more efficient by having the student determine what he needs to know, than by having the teacher build a model of his knowledge, present factual lectures, and test him on cases. However, the student might direct the ***teacher*** to do any of these in the process of actively directing his learning.

In contrast with the model developed in GUIDON, we are not using the knowledge acquisition heuristics to present information to the student, who must read facts and store them away. Instead, we focus on learning that occurs and is motivated by problem-solving failures. In contrast with GUIDON's original design, this is not a strategy for "filling in a knowledge base of the student." But again, the student might use these heuristics to request an ***orientation*** at particular times, just as the knowledge engineer applies these methods early in the knowledge acquisition process.

Just as for a knowledge engineer, the student's learning is failure driven, based on knowledge

of what he is trying to do (the form of an adequate solution) and what failures occurred. Specifically the learning procedure is as follows:

- Know what you are trying to do: constraints to satisfy (the form of a solution) and how to satisfy them (model-manipulation tasks).
- Detect possible failures (unsatisfied constraints) in the inferred, situation-specific model:
  - unable to test or refine a hypothesis;
  - unable to explain finding;
  - finding explained by two or more hypotheses;
  - two or more hypotheses explain exactly the same findings and evidence does not discriminate between them, or they explain findings uniquely;
  - situation-specific model hypotheses are not specific enough to select or construct action plans.
- Reason backward to say what task, if it had succeeded, would have prevented this failure, and what facts (the hypothesized gaps in the domain knowledge), if true or proved false, would allow the metarule to succeed.
- Prune alternative explanations using knowledge of what beliefs typically could be wrong or might be true, but which were not explicitly learned before.
- Ask the teacher questions to gain missing knowledge or validate hypothesized facts.

For example, referring to Figure 3-5, we consider the diagnostic constraint that every abnormal finding must be explained by the most likely hypothesis. We observe that the seizures finding is not explained. Relating this constraint to subtasks, we see that the HERACLES subtasks Test-Hypothesis (applied to acute bacterial-meningitis) and Process-Finding (applied to seizures) have associated inference rules (metarules) that would have satisfied this constraint if they had succeeded. Examining these metarules, we find that a domain rule linking seizures to acute bacterial-meningitis (among others) would enable one or more of the metarules to succeed. Stating this hypothesized fact as a question, the student would ask, "Could acute bacterial-meningitis cause seizures?*'

To emphasize again the limitations and complications of this model of learning:

- It is based on a fixed knowledge representation language and inference procedure;
- It requires making explicit the constraints of a solution and how they relate to diagnostic tasks and metarules;
- It requires a search procedure to work causally backward from failed constraints;
- It might require domain knowledge or domain-general knowledge about disorder processes in order to focus the search for plausible facts, if many possibilities are generated; and

- It may be necessary to relax the constraints imposed on the situation-specific model, given the pragmatic requirements of how the model will be used (e.g., the action plans it must discriminate between) and the inability to. confirm hypothesized facts (e.g., lack of scientific understanding of causal mechanisms).

Resolving these uncertainties and filling in details are good reasons for implementing the model as a simulation program.

4.2. Applying the model to tutoring

To apply this model to tutoring, we should place the student in an environment that is amenable to detecting failures, realizing gaps in knowledge, and hypothesizing and testing new facts.

Giving the student a problem to solve forces him to construct a situation-specific model. By making the model an explicit object for the student, we make the task less threatening, moving the problem outside of the student onto the screen in the form of a graph. Thus, we exploit the advantage of the KE as student-she is not wrong, it is the evolving computer program that is wrong. Yet, the KE naturally translates the program's deficiencies to deficiencies in her own understanding.

In addition to forcing the student to articulate his knowledge, we must teach him about the knowledge representation and inference procedure we expect him to apply. Assuming for a moment that this is an adequate model of human problem solving for routine problems, it is evident that our model of learning must recur. We are arguing that a specific learning process will lead the student to learn medical problem-solving facts, but how will the student learn the learning process itself?

The student must realize when solving the problem that he needs to articulate the process of diagnosis itself, in order to apply the learning process of working backward from a failure. Presumably, at this level, the tutor could actively prompt the student to lead him to criticize the solution and relate it to failed tasks. In particular, we must adjust our model of active learning to make clear the role of a teacher who prods the student, probes his understanding, and redirects his behavior.

What might go wrong with this instructional design? First, we face constant difficulties with level of detail. Explanations will have to carefully interweave specific and abstract descriptions so the diagnosis terminology is meaningful. Second, it is important to realize that NEOMYCIN has limited introspective ability to explain its design and reasoning. Human intervention may be necessary to explain the framework and its limitations.

Other problems include the possible need to articulate the learning process itself (to explain the tutor's advice) and to allow the student to provide his own explanations of the diagnostic

model for the tutor to respond to.

With this foundation we are now developing a tutorial program, named GUIDON2, to convey the NEOMYCIN model of problem solving and (yet to be implemented) active model of learning to a student. There are three phases in the tutorial interaction:

- **GUIDON-MANAGE** -- The student solves a diagnostic problem by abstracting his requests for data in terms of HERACLES' tasks (Rodolitz, 1987). For example rather than asking if the patient has a fever, he might give the command to Guidon-Manage, "Test the hypothesis of infectious-process." In one sense the student directs the diagnosis by providing the strategies to follow, while NEOMYCIN provides the tactics, using its metarules to apply domain knowledge. The result is a form of cooperative problem solving in which the student can rely on the program's domain knowledge, but must interpret the implications of the evolving solution and direct the problem solving. By being forced to use HERACLES' task language, the student is led to observe that each request for data has a more abstract characterization in terms of model building, and he learns the specific meaning of the diagnostic tasks encoded in HERACLES by observing what they do. In the most general sense, he learns that the diagnostic process has a recurrent structure, and he can start to rely on this when he gets stuck and is not able to proceed automatically.

- **GUIDON-WATCH** -- The student then watches NEOMYCIN solve the same problem. Knowing that NEOMYCIN follows a certain procedure, the student is now in a position to interpret the program's actions. That is, his experience with Guidon-Manage provides him with a vocabulary for explaining why NEOMYCIN requests patient data.

- **GUIDON-EXPLAIN** (proposed) -- The student and tutor then engage in a mutual explanation process in which they investigate significant differences in how the student solved the problem in the GUIDON-MANAGE phase and how NEOMYCIN solved the same problem in the GUIDON-WATCH phase. The tutor takes an active role of probing the student's understanding, while articulating its basis for criticizing a diagnostic solution so the student can realize his own failures and articulate his own missing knowledge.

By this instructional design we are teaching the process of knowledge engineering, not the product (the contents of a knowledge base). We are investigating how explaining, performing, and criticizing problem-solving behavior are interrelated and enhanced by the metacognitive ability to articulate the representation of diagnostic knowledge and how it is interpreted.

5. Related learning research

The model of learning described here relates educational research focusing on metacognition (e.g., (Schoenfeld, 1981)) to explanation-based learning (EBL) within AT (Mitchell, et al., 1986, DeJong and Mooney, 1986, Dietterich, et al., 1986). In this section, we consider briefly how our study extends machine learning research, and constitutes a model of failure-driven, explanation-based learning for non-formal domains.

The basic idea of explanation-based learning is that a surprising or unusual fact about the world is explained by the learner in terms of his a priori knowledge, making the fact explicit or more efficiently accessible for future use (Dietterich, et al., 1986). For example, the given information might be the features of an object, which could be used to infer that the object is a member of a certain class. This inference, which explains why the example satisfies the previously known definition of the class, takes the form of a proof, which is then generalized so that similar examples in the future can be more readily recognized.

Keller's explanation-based learning approach (Keller, 1986) resembles the model of active learning presented in this paper. His program uses contextual knowledge about how concepts are used, in order to formulate which concepts need to be learned. Keller's program incorporates knowledge about the "performance procedure and objective," which corresponds to the diagnostic procedure and constraints on the form of a diagnosis in HERACLES. The objective provides a criterion for determining the usefulness of a concept, called the ***operationality criterion.*** For most explanation-based learning, which focuses on deriving a relation that is already implicit in the knowledge base, the operationality criterion concerns efficiency of the inference procedure. That is, the goal of learning is to make the program able to solve a search problem that was previously too time-consuming.

The model of learning described here does not involve simply chaining together previously known facts and procedures, but conjecturing new facts or conjecturing the need for a certain type of knowledge. The operationality criterion is the description of a diagnostic solution, particularly its form as a causal model and how it will be used to select action plans (repairs). Learning is driven by failure to satisfy these constraints. It cannot automatically refine or generalize a previously known concept, as in previous explanation-based learning. Rather, the explanation of problem-solving failure is analyzed to determine the concepts or relations that could prevent the failure, which must then be confirmed or supplied in more detail by an expert-teacher. This analysis is a form of ***goal regression,*** a technique found in many EBL programs: We reason from the failed goal (unsatisfied diagnostic model constraint) to the task that could have satisfied the goal if it succeeded, back through failed metarules and failed metarule preconditions to other failed subtasks, eventually reaching ground facts about the world that are not in the knowledge base (e.g., the subtypes of a disease or what might cause

it).

This model of learning thus extends previous EBL in several ways:

- Goal regression involves reasoning through the problem-solving procedure itself (tasks and metarules), rather than a separate description of the procedure (as in Keller's program).

- The operationality criterion is described in terms of the form of a solution and how it will be used, rather than in terms of computational efficiency. (See (Dietterich, 1986) for discussion of the distinction between learning new knowledge versus chunking, compiling, or making computationally accessible what is already known.)

- Learning is based in explaining problem-solving failures, as detected by the program itself, not in explaining why a supplied example is correct. That is, the model involves determining what must be learned in order to properly solve problems, not just to increase problem-solving speed. Thus, this model bridges a gap between EBL and failure-driven learning (Schank, 1981, Kolodner and Simpson, 1984).

- The problem-solving procedure involves a schema-model of the world (the diagnostic relations of the knowledge base), which constitutes an incomplete theory, in contrast with the definitional model in domains like calculus (see (Clancey, 1986c) for further discussion).

-Generating plausible conjectures about missing knowledge is on the edge of learning research. Collins' early work in ***plausible reasoning*** suggests that metaknowledge about patterns in a knowledge base could be useful (Collins, 1978). Of special interest are the heuristics for conjecturing propositions that would lead to a consistent, parsimonious model of the world, if they were true. In this sense a knowledge base is not just a set of isolated statements, but a model providing a coherent, functional map of some system in the world. Examples from CASTER suggest that a KE's metaknowledge includes such general facts about causal models, which are generalizations about different domains in the form of recurrent terms and relations. For example, knowledge about manufacturing problems takes the form of a causal network relating abnormal structures (or substances) to abnormal functions (or processes), metaknowledge that goes well-beyond a representational description that is merely in terms of goals and rules [as in TEIRESIAS (Davis and Buchanan, 1977)] or states and causal-associational links [CASNET (Weiss, et al., 1978)].

Related research in knowledge acquisition includes:

- MORE (Kahn, et al., 1985) builds an initial knowledge base using a representation language and heuristics for improving a domain model that combine knowledge about the inference procedure and how to elicit knowledge from the expert.
- MOLE (Eshelman, et al., 1986) goes further by debugging a situation-specific model by comparing it to an expert's diagnosis, and makes clearer the nature of the inference procedure and knowledge elicitation strategies it relies on.
- The LEARNING APPRENTICE SYSTEM (Smith, et al., 1985) explains failures (described by an expert) in terms of errors in its rules, reasoning about the justifications of rules and possible kinds of errors. Of programs in nonformal domains, LAS is distinguished by attributing errors to assumptions that justify its causal reasoning. By reasoning about which assumptions are substantiated and which are likely to be wrong in certain contexts, the program engages in a sophisticated form of plausible reasoning.

Again, the major difference between this work and the active model of learning described in this paper is that we consider how a problem solver can ***detect his own failure*** to solve a problem and how he ***reasons through his inference procedure*** to explain how additional domain knowledge might have prevented the failure. The inference procedure is either implicit in learning heuristics used by the preceding programs or redundantly encoded in both the performance and learning programs. VanLehn's program, SIERRA (VanLehn, 1987) does reason through the inference procedure itself. However, its learning task is quite different: It is learning the inference procedure itself, not domain facts; its inference procedure is algorithmic, not heuristic; its domain is axiomatized; it learns by explaining how an expert solved the problem; and its learning is constrained by assuming that a sequence of examples is designed by the teacher to convey a single point.

The connection with Repair Theory (Brown and VanLehn, 1980) is particularly interesting. One wonders whether adults having trouble with subtraction would simply make up answers and continue by patching their incomplete knowledge in the manner described by Repair Theory, or would they attempt to articulate the nature of their knowledge deficiency as a question for the teacher if given a chance? Indeed, perhaps repairs might lead to conjectures about a correct procedure, which are tried and modified in solving later problems in an exam. An essential question is whether the subtraction inference procedure is articulated by the student in the course of recovering from failure or whether the procedure attributed by Repair Theory is just an abstraction that describes patterns in student behavior. The active model of learning has the advantage of describing problem solvers as active hypothesizers, who use general knowledge about the form of a solution and constantly learn while solving problems, which is closer to the model of SIERRA.

Finally, we might relate the active model of learning to comments by Bruner about the nature of learning (Bruner, 1983). He emphasizes that learning is "going beyond the

information given": "'Learning' is figuring out how to use what you already know in order to go beyond what you currently think." We have elaborated on this to show the advantages of describing "what you currently think" in diagnosis as a model of processes in the world. "What you already know" corresponds to the representation language, inference procedure, and support for beliefs.

Viewing thinking as a form of perception, learning how to think means developing strategies for perceiving. "Selective perception" is another way of describing the process of asking good questions. Crucially, Bruner emphasizes the importance of learning knowledge relations, what we have called the map language or the knowledge representation language: "The structure of knowledge permits us to grasp and retain and transform the world in a generative way not tied to the learning of details."

6. Conclusions

Generalizing a variety of learning heuristics used by **KEs,** we have described a model of learning, which provides a basis for designing a knowledge-acquisition program. We propose to incorporate this in an instructional program as a model of what we want to teach a student, a-means of evaluating and responding to his performance, and a model of his learning.

In constructing expert systems and incrementally improving our language for describing knowledge bases, we have adopted a view of learning and problem solving that is based in critiquing, improving, and applying models of the world. Model improvement, equated with learning, occurs after failure to model the world, that is, when a specific problem cannot be solved. A KE is continuously involved in this **task,** and we conjecture that other students could be taught the metacognitive knowledge to direct their own learning in a similar way. This learner-centered orientation should be contrasted with the prevalent concern of most intelligent tutoring research of attempting to understand the student by watching him solve problems, and with the equally strong concern in traditional educational research of question generation ***by the teacher*** (Bloom, 1956).

The analysis is complicated and violates many of our preconceptions about the design of teaching programs, because it is so different from what can occur in a typical crowded classroom. It focuses on the main strengths of knowledge-based tutoring: individualized instruction, driven by a single student's needs, and a simulation model of problem solving. The simulation model provides a basis for assisting and evaluating the student (in **guidon-**manage) as well as providing a basis by which the student can learning by watching (in **GUIDON-WATCH).** Throughout, we maintain the paradigm that a knowledge-based tutor must be able to do what it asks the student to do. Thus, we must include a formal model of **failure-**driven learning (as described here), as well as a model of learning by watching concurrently

developed in the ODYSSEUS program (Wilkins, et al., 1986)).

This paper is not a traditional AT research article, because it surveys and studies AT practice and existing programs without describing a new implemented program. In large part the novelty of the research direction requires this first paper as a statement of the model, so the difficulties and relation to other research can be explored first. Furthermore, theoretical connections tend to jump into place more quickly than we can complete the implementations.

Consider for example the ramifications of applying the active model of learning to a model of explanation. In an important sense the explanation program is the teacher responding to the active learner's question. The teacher might apply the learning model in reverse: Given a question from the student, what inference failure is the student coping with? What is the question asker trying to do? What representation language is he using to accomplish what tasks? What is his problem-solving procedure? This is a very difficult problem for the teacher, particularly when the student is experiencing a breakdown in his representation. Some of the difficulties were mentioned parenthetically in this paper as questions about the nature of knowledge and representation languages.

Consider further the justifications for the interview heuristics given in Section 3.2. What inherent problems in communication, deriving from the nature of knowledge and cognition, are these heuristics designed to cope with? Knowledge engineering methods go beyond designing notations for writing down what experts know; they touch on the very problems of the nature of knowledge itself.

For the moment it is sufficient to look back and observe that much progress has been made. We have come far from the simple capabilities of TEIRESIAS, which had no basis for detecting problem-solving failures on its own or relating them to MYCIN's model of the world, representation language, or inference procedure. TEIRESIAS could only say, "I couldn't conclude about the organisms that therapy should cover for," which is far from the well-focused question: "If I knew about the subtypes of chronic meningitis, I might be able to contrast it with brain abscess, and perhaps produce a single diagnostic explanation for both the headache and the double vision." This active process of learning, rooted in metacognitive knowledge, might be summarized by the aphorism, "A good question is more than halfway to a new understanding."

References

Bennett, J. ***ROGET: A knowledge-based consultant for acquiring the conceptual structure of an expert system.*** HPP Memo 83-24, Stanford University, October 1983.

Bloom, B. ***S. Taxonomy of Educational Objectives: The classification of educational goals.*** New York: David McKay Company, Inc. 1956.

Brown, J. ***S. Process versus product--a perspective on tools for communal and informal electronic learning,*** in ***Education in the Electronic Age, proceedings of a conference sponsored by the Educational Broadcasting Corporation, WNET/Thirteen,*** July, 1983.

Brown, J. S. and VanLehn, K. Repair theory: A generative theory of bugs in procedural skills. ***Cognitive Science,*** October-December 1980, ***4(4),*** 379-415.

Bruner, J. S. ***In Search of Mind: Essays in Autobiography.*** New York: Harper and Row Publishers 1983.

Bruner, J. ***S. Actual Minds, Possible Worlds.*** Cambridge: Harvard University Press 1986.

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. Proceedings of the Workshop on High Level Tools for Knowledge-Based Systems. Ohio State.

Clancey, W. J. GUIDON. In A. Barr and E.A. Feigenbaum (editors), ***The Handbook of Artificial Intelligence,*** chapter Applications-oriented AT research: Education, pages 267-278. William Kaufmann, Inc., Los Altos, 1982.

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

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

Clancey, W. J. ***Knowledge acquisition for classification expert systems,*** in ***Proceedings of ACM Annual Conference,*** pages 11-14, October, 1984.

Clancey, W. J. Heuristic Classification. ***Artificial Intelligence,*** December 1985, 27, 289-350.

Clancey, W.J. ***Viewing knowledge bases as qualitative models.*** KSL Report 86-27, Stanford University, 1986.

Clancey, W.J. From Guidon to Neomycin and Heracles in twenty short lessons (ONR Final Report 1979-1985). ***The AI Magazine,*** August 1986, ***7(3),*** 40-60.

Clancey, W.J. Qualitative student models. In Traub, J.F. (editor), ***Annual Review of Computer Science,*** pages 381-450. Annual Reviews, Inc., Palo Alto, 1986.

Clancey, W. J. ***Knowledge-Based Tutoring: The Guidon Program.*** Cambridge, MA: MIT Press 1987.

Clancey, W. J. Representing control knowledge as abstract tasks and metarules. (To appear in ***Computer Expert Systems,*** eds. M. J. Coombs and L. Bolc, Springer-Verlag, in preparation).

Clancey, W. J. and Letsinger, R. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. In Clancey, W. J. and Shortliffe, E. H. (editors), ***Readings in Medical Artificial Intelligence: The First Decade,*** pages 361-381. Addison-Wesley, Reading, 1984.

Collins, A. ***Fragments of a theory of human plausible reasoning,*** in ***Proceedings of the 2nd Conference on Theoretical Issues in Natural Language Processing,*** pages 194-201, Theoretical Issues in Natural Language Processing, July, 1978.

Crovello, T. and McDaniel, M. An Artificial Intelligence-Based Introduction to the Scientific Method. (Book in preparation).

Davis, R. and Buchanan, B. ***G. Meta-level knowledge: Overview and applications,*** in ***Proceedings of the Fifth International Joint Conference on Artificial Intelligence-77,*** pages 920-927, Massachusetts Institute of Technology, Cambridge, MA, August, 1977.

DeJong, G. and Mooney, R. Explanation-based learning: An alternative view. ***Machine Learning,*** 1986, ***1(2),*** 145-176.

Dewey, J. The process and product of reflective activity: Psychological process and logical form. In R.D. Archambault (editor), ***John Dewey on Education: Selected Writings,*** pages 243-259. Random House, Inc., New York, 1964.

Dietterich, T. G. Learning at the knowledge level. ***Machine Learning,*** 1986, ***1(3),*** 287-316.

Dietterich, T. G., Flann, N. S., and Wilkins, D. ***C. A summary of machine learning papers from IJCAI-85.*** Technical Report 86-30-2, Oregon State University, 1986.

Eshelman, L., Ehret, D., McDermott, J., and Tan, M. ***MOLE: A tenacious knowledge acquisition tool,*** in ***Proceedings of Knowledge Acquisition for Knowledge-Based Systems Workshop,*** pages 13-1--13-12, 1986.

Jackson, P. C. ***Introduction to Artificial Intelligence.*** New York: Petrocelli Books 1974.

Kahn, G., Nowlan, S., and McDermott, J. ***MORE: An intelligent knowledge acquisition tool,*** in ***Proceedings of the Ninth International Joint Conference on Artificial Intelligence,*** pages 581-584, Los Angeles, August, 1985.

Keller, R. M. ***Deciding what to learn.*** Technical Report ML-TR-6, Rutgers University, 1986.

Kolodner, J.L. and Simpson, R.L. ***Experience and problem solving: a framework,*** in ***Proceedings of the Sixth Annual Conference of the Cogtnitive Science Society,*** pages 239-243, Boulder, June, 1984.

Mitchell, T.M., Mahadevan, S., Steinberg, L.T. ***LEAP: A learning apprentice for VLSI design,*** in ***Proceedings of the Ninth International Joint Conference on Artificial Intelligence,*** pages 573-580, Los Angeles, August, 1985.

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based generalization: A unifying view. ***Machine Learning,*** 1986, I(I), 47-80.

Norman, D.A. ***Five papers on human-machine interaction.*** Technical Report ONR-8205, Center

for Human Information Processing, University of California, San Diego, May 1982.

Patil, R. S., Szolovits, P., and Schwartz, W. B. ***Causal understanding of patient illness in medical diagnosis,*** in ***Proceedings of the 7th International Joint Conference on Artificial Intelligence,*** pages 893-899, Vancouver, August, 1981.

Richer, M.H. and Clancey, W.J. GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. ***IEEE Computer Graphics and Applications,*** November 1985, ***5(11),*** 51-64.

Rodolitz, ***N. Tutoring for strategic knowledge.*** KSL Memo 87-38, Stanford University, June 1987.

Schank, R. C. Failure-driven memory. ***Cognition and Brain Theory,*** 1981, ***4(1),*** 41-60.

Schoenfeld, A. H. ***Episodes and executive decisions in mathematical problem solving.*** Technical Report, Hamilton College, Mathematics Department, 1981. Presented at the 1981 AERA Annual Meeting, April 1981.

Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. ***G. Representation and use of explicit justifications for knowledge base refinement,*** in ***Proceedings of the Ninth International Joint Conference on Artificial Intelligence,*** pages 673-680, 1985.

Thompson, T. and Clancey, W. J. A qualitative modeling shell for process diagnosis. ***IEEE Software,*** March 1986, ***3(2),*** 6-15.

Kurt VanLehn. Learning one subprocedure per lesson. ***Artificial Intelligence, 1987, 31(1),*** l-40.

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

Whorf, B.L. ***Language, Thought, and Reality.*** Cambridge: Technology Press of MIT 1956.

Wilkins, D.C., Clancey, W.J., and Buchanan, B.G. An overview of the Odysseus learning apprentice. In T.M. Mitchell, J.G. Carbonell, and R.S. Michalski (editors), ***Machine Learning: a Guide to Current Research,*** . Academic Press, New York, 1986.

Winograd, T. and Flores, ***C.*** F. ***Understanding computers and cognition: A new foundation for design.*** Norwood, NJ: Ablex 1985.