William J. Clancey

From GUIDON to NEOMYCIN and HERACLES in Twenty Short Lessons: ORN Final Report 1979-1985

Origins

The idea of developing a tutoring program from the MYCIN knowledge base was first described by Ted Shortliffe (1974). In fact, it was the mixed-initiative dialogue of the SCHOLAR teaching program (Carbonell, 1970) that inspired Shortliffe to produce the consultation dialogue of MYCIN. He conceived of it as a question-answer program in SCHOLAR's style, using a semantic network of disease knowledge. Shortly after I joined the MYCIN project in early 1975, Bruce Buchanan and I decided that developing a tutoring program would be my thesis project.

The GUIDON program was operational in early 1979. This review describes the key ideas in GUIDON and the important developments of the following six years as research continued under funding from the Office of Naval Research (ONR), the Defense Advanced Research Projects Agency (DARPA), and the Army Research Institute. The first three years were covered briefly in an earlier report (Clancey & Buchanan 1982). In general, only publications from this project are cited; many other references appear in the cited publications.

Overview: Introduction to the Programs

Figure 1 shows the relationship between programs we have constructed in the past six years, including MYCIN and EMYCIN, which served as the foundation.

The medical consultation system, MYCIN, was generalized to EMYCIN (van Melle, 1979). The tutoring system, GUIDON, was designed to work with any EMYCIN knowledge base (Clancey 1979a, Clancey & Letsinger 1984, Clancey 1982a).

NEOMYCIN, another medical diagnosis program, expands MYCIN's disease knowledge to include competing alternatives, for example, diseases that might be confused with meningitis. This provides an opportunity for teaching diagnostic strategy. MYCIN's strategy of exhaustive, top-down refinement is sufficient for the small set of diseases it knows about, but it is unrealistic for medical diagnosis in general. Using NEOMYCIN, we can convey the more complex processes of forming hypotheses, grouping competitors discriminating among competitors and reasoning causally. A second important idea in NEOMYCIN, distinguishing it from other expert systems, is that the inference procedure for diagnosis is represented in a well-structured language, separate from the medical knowledge. This facilitates explanation and student modeling.

HERACLES is the generalization of NEOMYCIN, standing for Heuristic Classification Shell (Clancey 1985a). By analogy with EMYCIN, we might say that HERACLES is "NEOMYCIN without the knowledge," but there is a big difference. We retain NEOMYCIN's diagnostic procedure; it is reused and adapted in new applications.

GUIDON2 is a set of tutoring systems that work for any HERACLES knowledge base; it is currently being developed with NEOMYCIN. In the GUIDON2 family of programs, we are exploring different forms of student and teacher initiative (Clancey 1984a).

Guidon: "Transfer of Expertise"

In GUIDON (See Figure 2), we held the MYCIN knowledge base constant and considered the additional knowledge about teaching that would provide a good tutoring system. We were especially interested in teaching from different knowledge bases using one program. This exciting

Abstract

I review the research leading from the GUIDON rule-based tutoring system, including the reconfiguration of MYCIN into NEOMYCIN and NEOMYCIN's generalization in the heuristic classification shell, HERACLES. The presentation is organized chronologically around pictures and dialogues that represent conceptual turning points and crystallize the basic ideas. My purpose is to collect the important results in one place, so they can be easily grasped. In the conclusion, I make some observations about our research methodology.

Bill Clancey is a senior research associate in the Stanford Knowledge Systems Laboratory, 701 Welch Road, Building C, Palo Alto, CA 94304

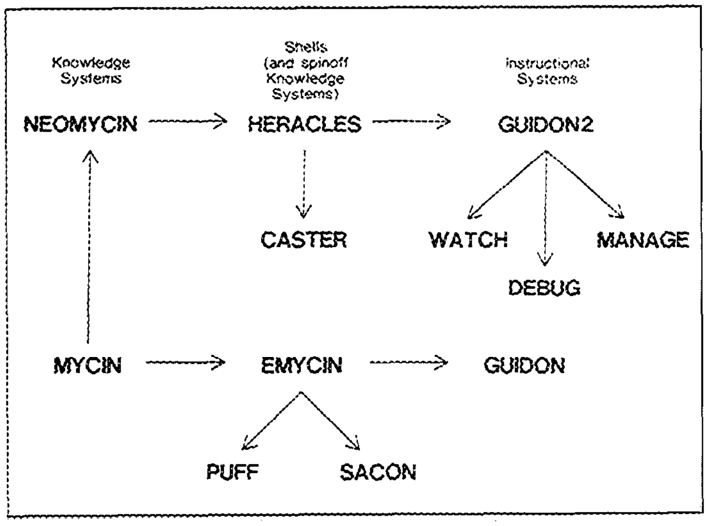


Figure 1. A Map Showing the Evolution of Research. NEOMYCIN, a reconfiguration of MYCIN, is also a medical consultation program. Research from both programs follows a parallel path: generalizing the system into a knowledge system shell, applying the shell to develop knowledge systems in other domains (PUFF, SACON, CASTER), and developing an instructional system compatible with any knowledge system developed from the shell. This paper describes the key ideas in GUIDON and the upper path of research.

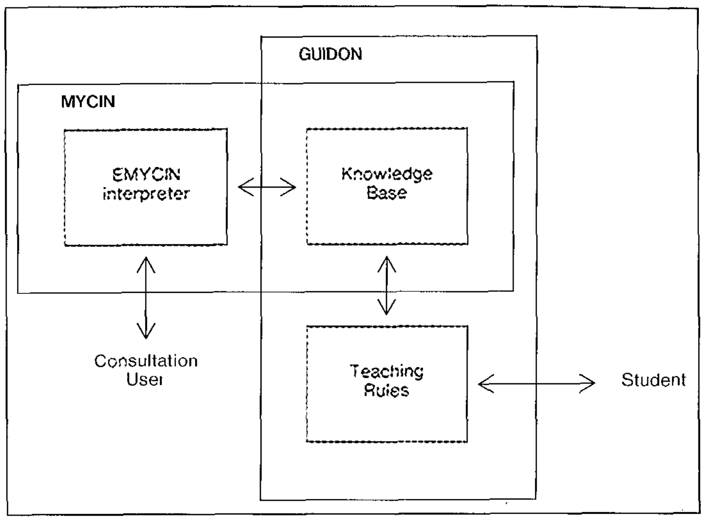


Figure 2. Guidon Teaches from MYCIN's Knowledge Base. The MYCIN knowledge base, combined with an interpreter for applying rules and interacting with a user, forms a consultation program The same knowledge base is interpreted by teaching rules for interacting with a student in a case-method dialogue, constituting the GUIDON instructional program. MYCIN's rules are ranked, relating them to years of medical experience (for modeling the student and selecting new material for the student to learn). Additional annotations indicate subtype and causal relations among rule clauses and relate the rules to a general description of the infectious process which is used by GUIDON to provide more concise explanations of MYCIN's reasoning. Within a few limits, GUIDON can discuss any case that MYCIN or any EMYCIN program can solve. In conventional computer-assisted instruction, a new program is written for each case.

idea was motivated by the EMYCIN design that allows putting in a different knowledge base and carrying on a consultation in a different domain. GUIDON's teaching knowledge is separate from the medical knowledge; so, it is reusable and adaptable to new applications. This is a significant advance over traditional computer-assisted instruction methodology that requires writing a new program for each case to be discussed. We now have two kinds of generality: First, this tutor can discuss any case that MYCIN can solve. Second, we can swap in a different knowledge base and discuss cases in another domain. The separation of the knowledge base from the procedures that interpret it is the important idea.

Discourse Procedures: Alternative Dialogues and Transitions.

Another successful aspect of GUIDON's design is the representation of the tutoring knowledge. This knowledge can be shown as a transition diagram, where each node represents a situation within a tutoring dialogue (See Figure 3). The program has a list of rules for reasoning about what to do at each step. For example, when GUIDON detects that a goal under consideration has been determined (from MYCIN's point of view), it selects from three alternative transitions: presenting a conclusion, presenting a summary, and asking the student to make a hypothesis (Clancey 1979b).

Each of these transitions is encoded by rules called tutoring or t-rules, numbering about 200. We built the system very much like a traditional expert system, running cases and incrementally modifying the t-rules. When the program said something inappropriate, we modified t-rule conditions to change when that kind of remark would occur. Similarly, GUIDON sometimes missed an opportunity to say something interesting. For example, if a fact can be inferred by definition, there is no need to go through a long dialogue, gathering data and forming hypotheses and so on; so we added t-rules to deal with this case, leading the program to give MYCIN's conclusion or to ask for the student's conclusion, depending on the model of what the student knows and the goals for the dialogue.

This works rather well, though it lacks a theoretical foundation. Arbitrary strategies are encoded in the tutoring rules. Building on the t-rule idea, Beverly Woolf has added a hierarchical structure to the alternative dialogues, couched in the terminology of discourse analysis. This represents in a more principled way the choices the program is making. (See Woolf & McDonald 1984.)

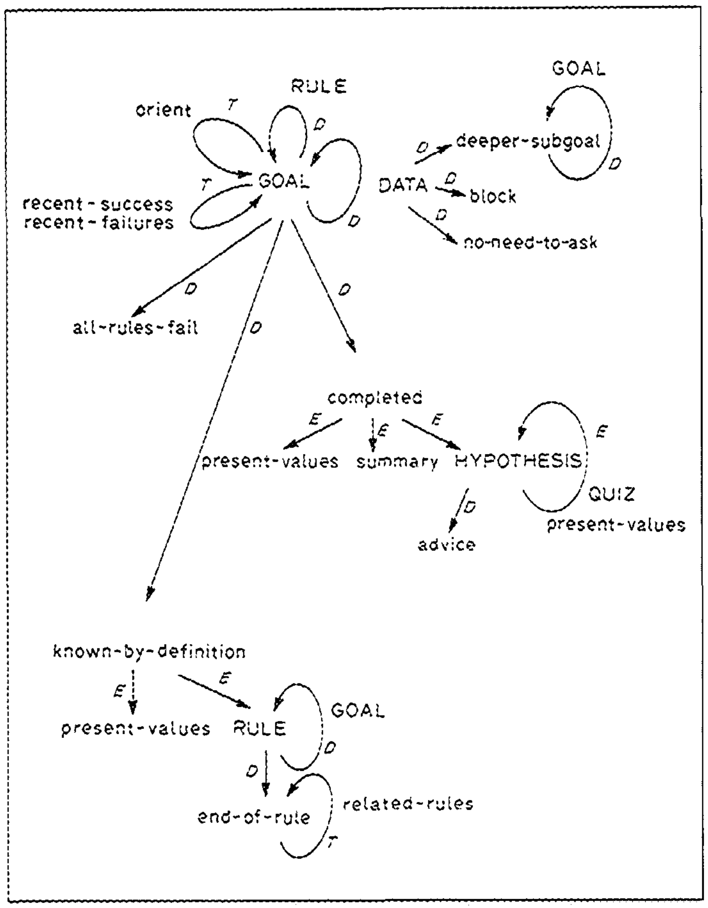


Figure 3. Dialogue Transition Diagram. Each node stands for a situation in a case-method dialogue, represented in GUIDON as a stylized procedure of ordered rules or a rule set, totalling 200 rules in 40 situations. For example, in pursuing a goal explicitly agreed upon by student and program, the student can request more case data. GUIDON can recognize the data as relevant to a subgoal, provide it as a set of related information (block), or determine that there is no need to ask (perhaps because the requested data can be inferred from known information). Arrows that loop back indicate that a situation may occur iteratively or recursively. For example, several related rules might be presented after a given rule is discussed. The italicized labels indicate the basis for a transition: Economy, Domain logic, and Tutoring goals.

Overlay Model: Evaluating a Student Hypothesis

Perhaps the most interesting reasoning in GUIDON involves evaluating a student's partial solution (See Figure 4) (Clancey 1979c). In this example, the student says that the organisms causing the infection could be *Diplococcus*, *Pseudomonas*, or *Neisseria*. The program looks at MYCIN's rules and sets up a consistent mapping. It uses double evidence, a history of interaction with the student, and a measure of rule difficulty to construct a consistent model.

For example, suppose the student mentions *Neisseria.*

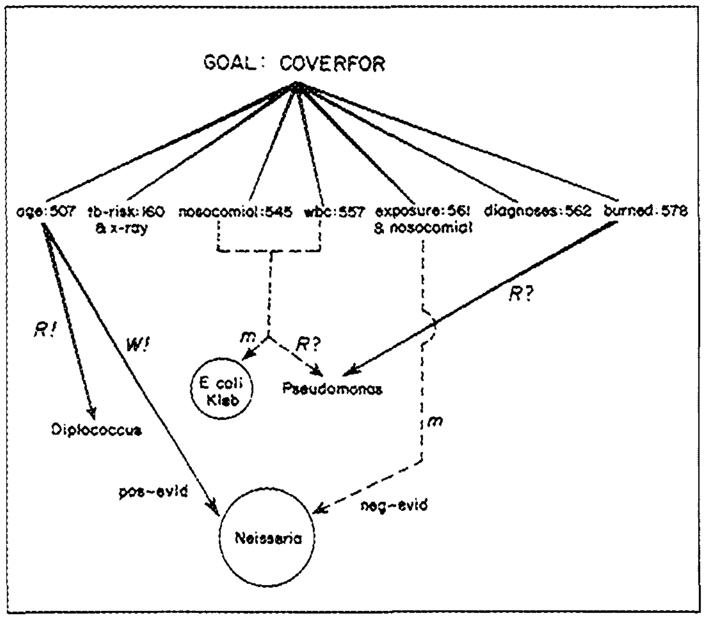


Figure 4. A Student Model Constructed by the Overlay Technique. The student states that the organisms causing the infection might be *Diplococcus, Psuedomonas,* or *Neisseria.* MYCIN rules that conclude about the organisms causing the infection are shown with associated patient data For example, rule 507 states that if the patient is between 15 and 55 years old, then *Diplococcus* and *Neisseria* are organisms that therapy should cover. Circled values are missing from the student's hypothesis (for example, *E.coli*) or wrongly stated (for example, *Neisseria*). Dotted lines lead from rules the student probably did not use. m = evidence link that the tutor believes is unknown to the student. R and W = links to the right and wrong values the tutor believes are known by the student; ! = a unique explanation; the tutor knows of no other evidence at this time. ? = questionable, the tutor is uncertain about which evidence was considered by the student. For example, R? means that the student stated this value, it is correct, and more than one MYCIN rule supplies evidence for it.

If MYCIN's rules argue for and against this hypothesis, then the student might know the positive evidence, but not the negative evidence. GUIDON concludes in a similar way that *Pseudomonas* is believed by the student because the patient is burned—but not because of the white blood count (WBC) or because the infection occurred in the hospital (nosocomial)—given that the student didn't mention the two other diagnoses associated with this evidence (*E.coli* and *Klebsiella*). It's a straightforward, logical analysis, demonstrating the value of production rules for indexing how facts are concluded and used in a program.

The Sacon Tutorial: Experimenting with other Knowledge Bases

Figure 5 shows an excerpt from a dialogue with GUIDON using the SACON knowledge base. It graphically demonstrates the value of keeping GUIDON's tutoring knowledge general: The same tutoring program can interpret SACON's rules and discuss a structural analysis problem with a student.

```
We must now determine whether the shape of the floor section (SUB-STRUCTURE-1) is one of: the surface shapes.
Here is some relevant data you could have asked for: the geometry of the floor section is planar, the modelling dimensionality of the floor section is 2, and continuum is one of the constructions of the floor section.
What does this tell you about the shape of the floor section?
** HELP
Can you conclude that the shape of the floor section is (select from the following):
(1) beam—suggestive evidence (.62)
(2) semimonococque—suggestive evidence (.62)
(3) not shell—strongly suggestive evidence (-.95)
(4) plate—with certainty (1.0)
** 4
Yes, that's right (see RULE100).
```

Figure 5: Excerpt of GUIDON Tutorial Using SACON Knowledge Base. GUIDON encounters an incomplete subgoal in a rule that it's trying to discuss with the student. A t-rule in the procedure for discussing an incomplete subgoal finds that the subgoal can be inferred by a definitional rule and then invokes the procedure for discussing definitional rules. GUIDON gives the student new information (the geometry, dimensionality, and construction of the floor section), and then asks him if he can now infer the shape of the floor section. The student asks for help, and GUIDON converts the question into a multiple choice. Reasoning about the current problem state, text generation, and quiz construction and evaluation are all accomplished by general t-rules that were originally developed in the context of a medical diagnosis dialogue.

This interaction plausibly captures some of the behavior we'd like to see in a teaching program. It was produced entirely by t-rules that were written for medical examples and then just plugged into SACON. It took about an hour to make it all work, with a few modifications to cope with syntactic variations in SACON's rules. For further discussion and an example from PUFF, see (Clancey 1979a, Clancey 1982a).

Inadequacy of Mycin: Implicit, Nonpsychological Strategy

We now consider the analysis that led to NEOMYCIN. What problems arise in using MYCIN for teaching?

Figure 6 shows an excerpt from an experiment with GUIDON; this was a pivotal example for me. GUIDON indicates that the the age of the patient, 34, is not evidence for *Neisseria.* Yet, a rule in the knowledge base says, "If the age of the patient is between 15 and 55, then Neisseria is one of the organisms." I was rather surprised. This rule is consistent with the student's hypothesis and justification.

The problem is that some of the information in the premise of this rule is still unknown, so MYCIN can't apply the rule. Specifically, there is no indication that the age of the patient is causally related to *Neisseria* and that the age would be sufficient in itself to suggest this conclusion. GUIDON has no way of knowing that one of the clauses is more directly associated with the conclusion than any other clause. To make this clear, consider another rule: "If the age of the patient is greater than 17 and the patient is an alcoholic, then *Diplococcus* might be causing the infection." Considering this rule and knowing only that the patient is 34 would not make you think of *Diplococcus.* Here the age clause controls the application of the rule, preventing the program from asking whether a child is an alcoholic. The causal relation is between alcoholism and *Diplococcus.*

The student's knowledge and the procedure being used are very different from MYCIN. The student has probably formed a hypothesis just hearing the age of the patient and some tentative information (not shown in the excerpt) that suggests meningitis. MYCIN will only conclude *Neisseria* when, from its point of view, it has exhaustively considered the evidence for meningitis and considered whether it is bacterial and so on. MYCIN does a top-down search through the set of diseases, but the student has "triggered" meningitis from just partial information, with no direct evidence for an infection or bacterial infection at all.

To properly respond to the student, we would have to represent the association between age and *Neisseria* explicitly and separate out the search procedure. However, to recognize what strategy the student is following, we'd have to encode a different strategy, expressing why it makes sense to think about *Neisseria* just knowing the age and some tentative evidence for meningitis. The very idea of a hypothesis is foreign to MYCIN.

The Idea of Strategy: From "tracing a parameter" to "focusing on a hypothesis"

Figure 7 and Figure 8 illustrate that at a certain level MYCIN's reasoning is arbitrary, lacking the focus on hypotheses we find in people. People group their questions logically; they don't jump around without reason. However, MYCIN does not focus on a particular hypothesis as it goes down through its (implicit) tree of diseases. When it considers types of meningitis or organisms, the types are considered arbitrarily, based on the order in which rules were entered into the program. The program proceeds systematically from infection to meningitis to bacte-

```
What are the organisms which might be causing the infec-
tion?
>> NEISSERIA-MENINGITIDIS
What facts about this case tell you that Neisseria-meningitidis
is one of the organisms which might be causing the infection?
>> AGE
>>
- > The fact that the age of Pt538 is 34 years is not evidence
for the conclusion that Neisseria-meningitidis is one of the
organisms which might be causing the infection.
Do you have a question?
**
```

Figure 6: Problematic Excerpt of GUIDON Tutorial Using MYCIN Knowledge Base. The student offers to state a partial solution (student input follows ">>"). The program responds by rephrasing the current topic as a question, "What are the organisms which might be causing the infection?" When the student says *Neisseria,* GUIDON checks and finds that MYCIN has made no conclusion at all up to this point. A t-rule prompts the student to justify his hypothesis. The student says that he is considering the age of the patient. It's a lost cause for the student, however; whatever he says next GUIDON will reply, "No, that's not sufficient," because MYCIN has made no conclusion. At the final prompt, the student can review the available data and MYCIN's reasoning, if desired. In fact, the student's hypothesis is reasonable, but GUIDON would need to know how MYCIN's rules are constructed and a different model of reasoning to understand why the student did something different.

rial meningitis to organism, but the process is unordered at each level of refinement with regard to children. This is because the goals that MYCIN pursues are always more general than the conclusions in the rules being applied. In order to teach a procedure to a student and to recognize what the student is doing, we need a program that will deliberately focus on particular diseases and that will be able to articulate its focusing principle.

This analysis of MYCIN was directly inspired by a study of strategy by Brown (Brown, Collins, & Harris 1977). He points out that a problem solver does not apply algebraic operators randomly when simplifying an equation; there is some logic behind each choice, describing a line of reasoning. Applying this analysis to MYCIN, I understood for the first time how a strategy reasons about operators or problem-solving methods, focusing their application. In MYCIN, a rule corresponds to an operator, and problem solving involves some strategy for selecting which rule to apply. Specifically, diagnostic reasoning is usefully controlled by *focusing* data requests and hypothesis testing.

From this perspective, it can be seen that describing strategy only in terms of domain rule ordering, as in Davis's original conception of metarules, is inadequate. The problem is that there is an implicit, undisciplined mapping between medical knowledge and MYCIN's parameter-value language. For example, if MYCIN's diseases were all represented by individual parameters (rather than by a general parameter called "coverfor" with organisms as values), then the normal back-chaining process would make the reasoning focused. Thus, a strategy can be implicitly encoded in the relation between parameters and their values. Recalling the age-alcoholism example, a strategy is already implicitly coded in the ordering of rule clauses. Before metarules can be written to systematically control domain rules according to a hypothesize-and-test strategy, conventions must be established for distinguishing between data and hypothesis parameters and consistently encoding causal and subtype relationships among them.

The Tetracycline Rule: Structure, Strategy, and Support

This brings us to about 1980 when I studied MYCIN's 400 rules to determine how they might be reconfigured for use in teaching. Up to this point, in constructing GUIDON, only limited annotations had been added to the original rule set. Now any change at all would be allowed.

Early on, I developed a framework that turned out to be very useful in protocol analysis. In this framework, explanations are analyzed according to *knowledge roles*, how knowledge is used in relation to other knowledge (Clancey 1983a) (See Figure 9):

- The *heuristic rule*: A relation between data and diagnoses or therapies
- *Structure*: Subsumption relations among data, diagnoses, and therapies
- *Strategy*: The procedure for applying rules
- *Support*: The justification for rules

As an example of structural knowledge, you might think of "SSS" when trying to remember this framework.

To understand the different kinds of knowledge here, suppose that you have a patient who is four years old. You would probably say, "Well, I'm not going to prescribe tetracycline." However, that's not a very good model of how a physician reasons; this would be a strange conclusion to make right at the beginning. Just as there is a logic for requesting data and for focusing on hypotheses, there is a logic for making assertions given available information. All together, we call this logic the *inference procedure.* (In the literature, it is also called *control knowledge* or, more specifically, *diagnostic strategy.* To emphasize the reasoning about control alternatives, the term managerial strategy or metastrategy is also used, particularly in the education literature [Schoenfeld 1981]).

By strategy, I mean the general goal that leads the physician to remember a heuristic rule. For example, when

```
32) Does J. Smith have a history of NEUROSURGERY?
** NO
33) Does J. Smith live in a crowded environment?
** NO
34) Do you suspect recent alcoholic history in J. Smith?
** NO
35) Is the meningitis a hospital-acquired infection?
** YES
36) Is J. Smith's clinical history consistent with
EPIGLOTTITIS?
** NO
37) Is J. Smith's clinical history consistent with
OTITIS-MEDIA?
** NO
38) Has J. Smith ever undergone splenectomy?
** NO
39) Is J. Smith a burn patient?
** YES
```

Figure 7: Sequence of Data Requests from MYCIN Consultation.

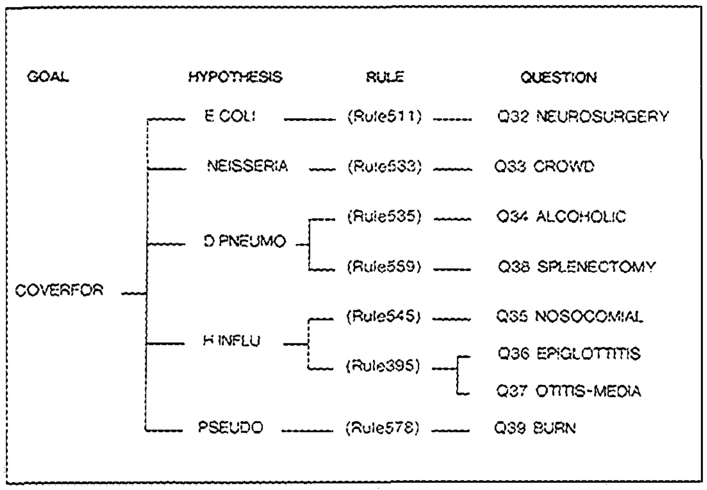
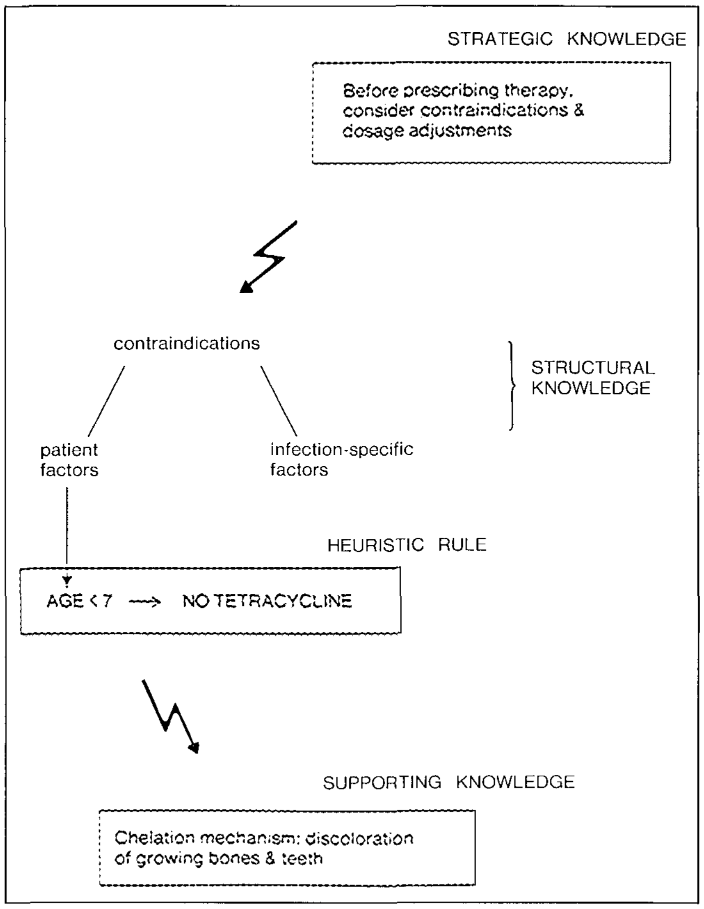


Figure 8. Relating MYCIN's Data Requests to Organism Hypotheses. MYCIN's questions, shown in Figure 7, have been reordered according to the hypotheses that motivate them. For example, question 33 about living in a crowded environment is asked in order to apply rule 533, which concludes *Neisseria.* All of the questions pertain to the same goal — determining what organisms therapy should cover — but the rules conclude about different organisms. Neither the sequence of rule applications nor the questions are sorted by organism Questions 34 and 38 pertain to *Diplococcus-pneumoniae*, with three intervening questions pertaining to *Hemophilus-influenzae.* Thus, in pursuing a goal, MYCIN's reasoning is unfocused at the level of possible values for the goal, in this case organisms that might be causing the infection.



STRATEGIC KNOWLEDGE
Before prescribing therapy, consider contraindications & dosage adjustments
contraindications
STRUCTURAL KNOWLEDGE
patient factors
infection-specific factors
HEURISTIC RULE
AGE < 7 ⟶ NO TETRACYCLINE
SUPPORTING KNOWLEDGE
Chelation mechanism: discoloration of growing bones & teeth

Figure 9. Analysis of Knowledge Relating to MYCIN's Tetracycline Rule. The rule states, "If the patient is less than seven years old, then remove tetracycline from the list of drugs under consideration." Relation of age to other contraindication factors (such as whether the patient is pregnant), justification for the rule, and time when it would be considered are relevant to explaining this rule, but are not represented in MYCIN. Making explicit this structural, support, and strategic knowledge enhances our ability to understand and modify MYCIN

is it important to remember not to prescribe Tetracycline?

Obviously the physician must take this into account when prescribing therapy.

By structural knowledge, I mean the relations by which heuristic rules are indexed and subsequently controlled. In general, this involves categorizing the facts the rules use (for example, patient factors) and the facts they conclude about (for example, therapies).

By support knowledge, I mean the justification for the rule. Why wouldn't you prescribe tetracycline to someone who is less than seven years old? Here we have a chemical process, a chelation mechanism, that results in the molecule binding to the growing teeth and bones, and a social consideration that attests people don't want to have discolored teeth. This is a very interesting justification because it shows that giving tetracycline might save the patient's life, even though it might have an undesirable side effect. That's important to know if tetracycline is the only drug available. It's a nice example of why it's useful to know the justification of a rule—so you can violate the rule and know what the consequences will be.

Generally, this figure suggests a framework for understanding an expert's explanations. When I ask a physician who is solving a problem, "Why did you ask that question?" I classify the answer into one of these categories. If the physician tells me, "Well I'm not going to prescribe tetracycline because the age is less than seven," I am being told what assertions were made from given information (that is, a heuristic rule). If I ask the physician why and I get an explanation having to do with chelation, then I'm being given the justification for the assertion (that is, support). If the physician says "This is just one of the contraindications I'm going to consider," then I'm being told about the organization of his knowledge, the categories used for focusing (that is, structure). Finally, if the physician tells me when contraindications are considered and how each type is considered, then I'm getting the inference procedure (that is, strategy). I tried to consistently apply this analysis when working with physicians, particularly to focus their explanations on strategy and avoid the bottomless pit of support explanations.

NEOMYCIN research focuses on representing strategy and structure because this is the deficiency of GUIDON we most want to improve. We also sense that structure and strategy are at the top of a pyramid of knowledge and are more limited in nature. A research effort focused on them is attractive because this knowledge conceivably might be carefully and exhaustively explored.

The Beckett Tapes: An Articulate Teacher

In 1980, Reed Letsinger and I worked with Tim Beckett, M.D., who was recommended by Ted Shortliffe and who turned out to be a rather fortunate choice. Beckett was known at Stanford for being a good teacher. He could articulate general principles for reasoning very well. He didn't just say what it is you should ask about or what your conclusions should be—he was able to speak in general terms about how you should think.

We taped interviews and classroom interactions, and transcribed and studied them (Clancey 1984b). In one interaction, Beckett interrupts a student who is examining a patient played by another student:

> When you ask these questions about whether gargling makes it better or worse, or whether it's better certain times of the day, are you thinking about how that's going to help you move down different diagnoses? ... ask a couple of general questions maybe that could lead you into other areas to follow up on, rather than zeroing in.

Note the absence of medical terms in his strategic advice. Again:

> We're talking at the top of infections, but before we go down infections, are there any other things you can think of? The mistake you don't want to make is leaving out the important things on top.

We repeatedly heard these general statements—move down different diagnoses, ask general questions, don't leave out important things on top. These were the strategic gems—better than I could have expected—that would allow us to construct NEOMYCIN. Essentially, I saw the opportunity here for a program that would talk procedurally about these operations: Moving down different diagnoses, asking general questions, not leaving out important things at the top. This procedure is separate from the medical knowledge, describing how the medical knowledge is searched. That is, the statement of strategy does not directly mention domain terms; it is abstract.

In Beckett's explanations, we see regular switching back and forth between the concrete situation and a generalization:

> Ask it very generally, like "Have you had any major medical problems, or are you on any medication?" Those types of general questions are important to ask early on because they really tell you how soon you can focus down.
>
> You have to think of some of the common things, but at the same time you have to think of some of the serious things that may not be common. What is a serious infection that can get in your throat?

This last example shows most clearly my model of inference in NEOMYCIN.

Refining the diagnosis and thinking of some of the common things, the physician looks into the domain model and asks, "What is a serious infection that can get in the throat?" and "What are some of the common things that could cause it?" This is how the metarules in NEOMYCIN work.

As confirmation of the potential effectiveness of Beckett's approach, we analyzed his best student's reasoning. The student obviously followed the procedure Beckett articulated in class. Of course, not all students would necessarily find Beckett's teaching approach to be useful, but we had an existence proof and clear statements of at least one diagnostic procedure, so we wrote the approach down.

About this time, we also had the first glimmer of how an explicit procedure could help a student learn relevant medical knowledge. When I had Beckett present problems to me, I often lacked the medical knowledge to carry out the procedure. However, knowing the procedure, I found that I could ask reasonably intelligent questions: "I know I should be thinking about some of the serious and common causes of this disease, but I don't know what they are." This has evolved into our version of explanation-based learning (see The Situation-Specific Model: From a Diagnosis to an Exploration). We also applied the procedure to an analysis of Beckett's interruptions of students: Given this model of his reasoning, could we use it to infer his strategy for interrupting students and providing assistance? The most telling example, occurring just before Beckett asks the question about sore throats shown here, is analyzed in (Clancey 1984c).

Neomycin: Separating the Medical Knowledge from the Diagnostic Procedure

Figure 10 shows the architecture of NEOMYCIN, illustrating the idea of separating the diagnostic inference procedure (control knowledge) from the medical knowledge. Crucially, both are represented in well-structured languages so that they can be reasoned about by the explanation, knowledge-acquisition, and student-modeling programs (Clancey 1983b).

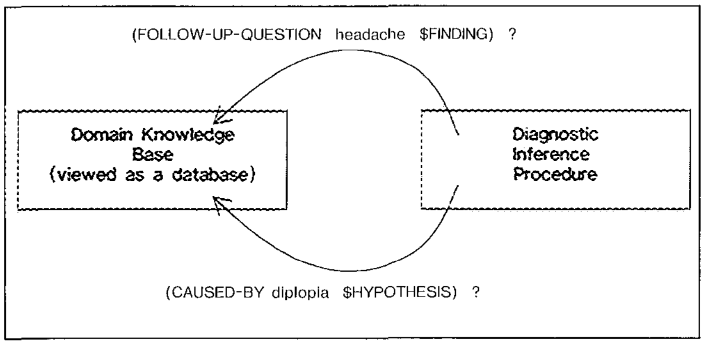

(FOLLOW-UP-QUESTION headache $FINDING) ?
Domain Knowledge Base (viewed as a database)
Diagnostic Inference Procedure
(CAUSED-BY diplopia $HYPOTHESIS) ?

Figure 10. Architecture of NEOMYCIN. An inference procedure queries the knowledge base, relating findings and hypotheses to one another in order to make a diagnosis. For example, given that the patient has diplopia (double vision), the program asks the knowledge base what could cause it One or more hypotheses might be returned, which the inference procedure will proceed to discriminate, test, and refine, making further inquiries about disease and symptom relations.

Davis's conception of metarules for expressing strategy inspired this design. However, TEIRESIAS's metarules compose domain facts with procedure, just like MYCIN's rules (Clancey 1983a). NEOMYCIN's metarules mention no domain terms. Moreover, they constitute a coherent procedure that completely controls every data request and every inference; so there is no back chaining of rules at all.

As is apparent in Beckett's generalizations, we can think of this procedure as "asking questions of the domain model." The language of relations used in metarules corresponds to the propositions in the knowledge base. These relations impose a classification on domain terms. This is what I called structural knowledge in the tetracycline analysis.

Given a hypothesis, the program asks, "What is a common cause of this disorder?" The program then looks up this relation in the knowledge base. In this sense, the inference procedure is interpreting the domain model. If we compiled the procedure—instantiating and composing it with respect to a particular knowledge base—we would get something very similar to MYCIN's rules. In making this abstraction, stating these general rules, I'm not claiming that people reason through general statements every time or even realize that these patterns exist. In particular, reasoning categorically probably involves automatic processes of memory. Some distinctions, such as considering causal prerequisites of diseases before effects, might be regularities that the physician does not consciously realize (Clancey 1984c).

I now believe that these domain relations are in large part what we want to teach students, as generalizations, to help them learn about new diseases. In describing how to focus reasoning, we are indirectly saying how knowledge should be practically organized. For example, we say, "You should think in terms of common causes and serious causes." That is much more informative than saying, "You should form a hypothesis" or "You should reason forward." We hypothesize that the procedure is automatic once you have the knowledge. A medical student might not have to be told to refine hypotheses, but he[1] has to be taught the subtypes of fungal meningitis.

The Disease Taxonomy: Searching an Abnormal Process Classification

There are several dimensions for describing NEOMYCIN's reasoning: psychological aspects of memory and attention, AI representation and control techniques, and aspects of medical causal reasoning. Figure 11 provides one perspective in which these dimensions come together.

The main part of the knowledge base is a taxonomy of diseases or, more generally, a classification of abnormal processes. Each disease describes a process, something that has happened to the patient in the past, accounting for the set of observed manifestations. In general, there can be many different taxonomies, orthogonal and tangled.

How do we know that a given taxonomy is complete? This important question did not explicitly arise in MYCIN research because we didn't isolate the disease taxonomy as a separate object of study. We now hypothesize that the physician's diagnostic classification, particularly its level of specificity, depends on how it will be used. The physician is not involved in scientific research here; what goes into the taxonomy is based on distinctions useful for selecting therapy. For example, NEOMYCIN makes no attempt to determine precisely which type of viral meningitis the patient has. The reason is that they're all treated the same—with a lot of aspirin and orange juice—and it is irrelevant to resolve the cause any further. Thus, NEOMYCIN's

[1]Masculine expressions in this article are used as generic terms No bias is intended

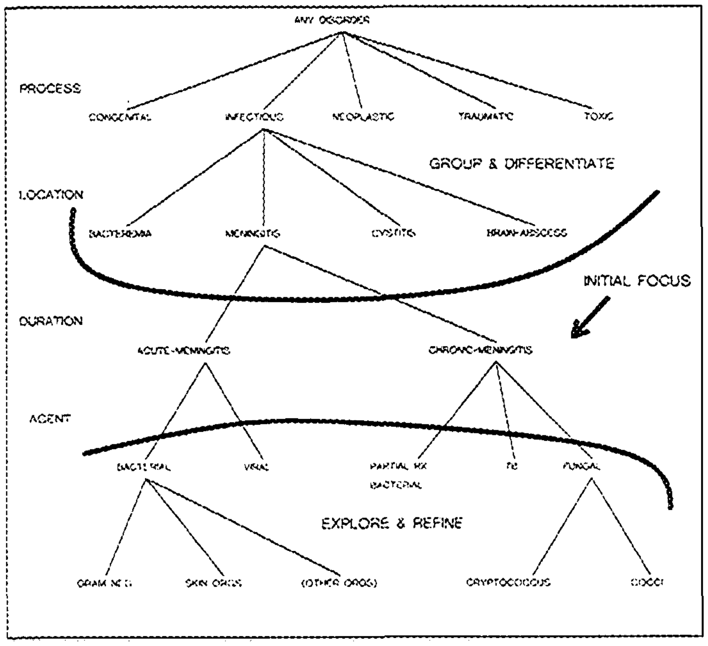


Figure 11. Looking Up and Looking Down in Diagnostic Search. Disease knowledge is represented as a taxonomy of processes. At the highest level are internal aberrations in structure building or maintenance (for example, congenital diseases) and processes involving environmental interaction (for example, infection, trauma). Processes are specialized here by location, temporal extent, and specific agent. The taxonomy is overprinted to show hypothetically how it might be searched. Initial information—chief complaints—triggers some hypothesis, shown arbitrarily here in the middle of the disease taxonomy. Two operations follow: (1) looking up, thinking of the high-level categories and discriminating among them (GROUP-AND-DIFFERENTIATE) and (2) looking down to refine hypotheses when distinctions are important for selecting therapy (EXPLORE-AND-REFINE). This is to be contrasted with an exhaustive, top-down search, which a large knowledge base makes impractical.

disease taxonomy deliberately remains a partial model of abnormal processes within this area of medicine.

Another part of the knowledge base, the causal network, is discussed in the context of CASTER (See CASTER: From Disease to Abnormal Substances and Processes.)

The Diagnostic Procedure: Search Operators and Constraints

The overall diagnostic strategy or inference procedure is a program consisting of a set of subprocedures as shown in Figure 12.

Each procedure is represented as a set of ordered and controlled conditional statements called *metarules.* Rules provide a uniform, well-structured language. Although experienced programmers can read a LISP encoding of the diagnostic procedure easily enough, it is difficult to write a program that can understand arbitrary LISP code. Too much of the design is implicit and not available for explanation. Therefore, we devised a highly structured representation, organized around the idea of rule sets, with every "loop" encoded as a separate task (subprocedure) and the control of rules stated declaratively (simple vs. iterative, try-all versus stop-on-success). Each task has a typed focus (argument), local variables, and an explicit "end condition" (equivalent to the "while" or "until" condition of a loop). Making every program statement a rule facilitates interpreted control, annotation, and record keeping.

The overall design is similar to LOOPS, which evolved at the same time as NEOMYCIN. However, NEOMYCIN's metarules use variables, rather than domain terms. Also, the end condition, inherited by task invocation, enables a procedure anywhere on the current stack to regain control, either because its goal is completed or there is reason to reconsider how its subgoals are being accomplished. Figure 13 shows the flow of control in terms of focus changes.

In writing down the diagnostic procedure as rules, we are following the same methodology used in developing MYCIN and GUIDON. With the knowledge expressed in a disciplined way, it now becomes possible to study patterns and to consider how the knowledge could be derived. Such implications are too numerous to recapitulate here. The interested reader will find the metarules listed in (Clancey 1984c), with a discussion of the procedure in terms of *operators* and the cognitive, social, mathematical, and case population *constraints* implicit in the rules. The next section considers the procedure as a *grammar.*

Image and Odysseus: Parsing the diagnostic process

Given the abstract nature of the tasks and metarules, they can be viewed as a kind of grammar for parsing a problem solver's sequence of requests for data. Such an analysis is shown in Figure 14, the picture I had in the back of my mind in about 1980 when I wanted some way for GUIDON to reason about what a student was doing. An interpretation of a student's partial solution provides a good basis for assisting him when he doesn't know what do next. Such an interpretation is also a source of information for relating a student's explicitly stated diagnosis to a model of his domain knowledge. As a contextual analysis, it potentially shortens the interactive dialogue that might be necessary to confirm the student's understanding.

Bob London, David Wilkins, and I have been developing student-modeling programs with the common goal of using NEOMYCIN's diagnostic procedure to interpret a sequence of student requests for data. London has followed a top-down approach in the IMAGE program (London & Clancey, 1982); Wilkins's ODYSSEUS program uses exhaustive, bottom-up reasoning (Wilkins, Buchanan, &

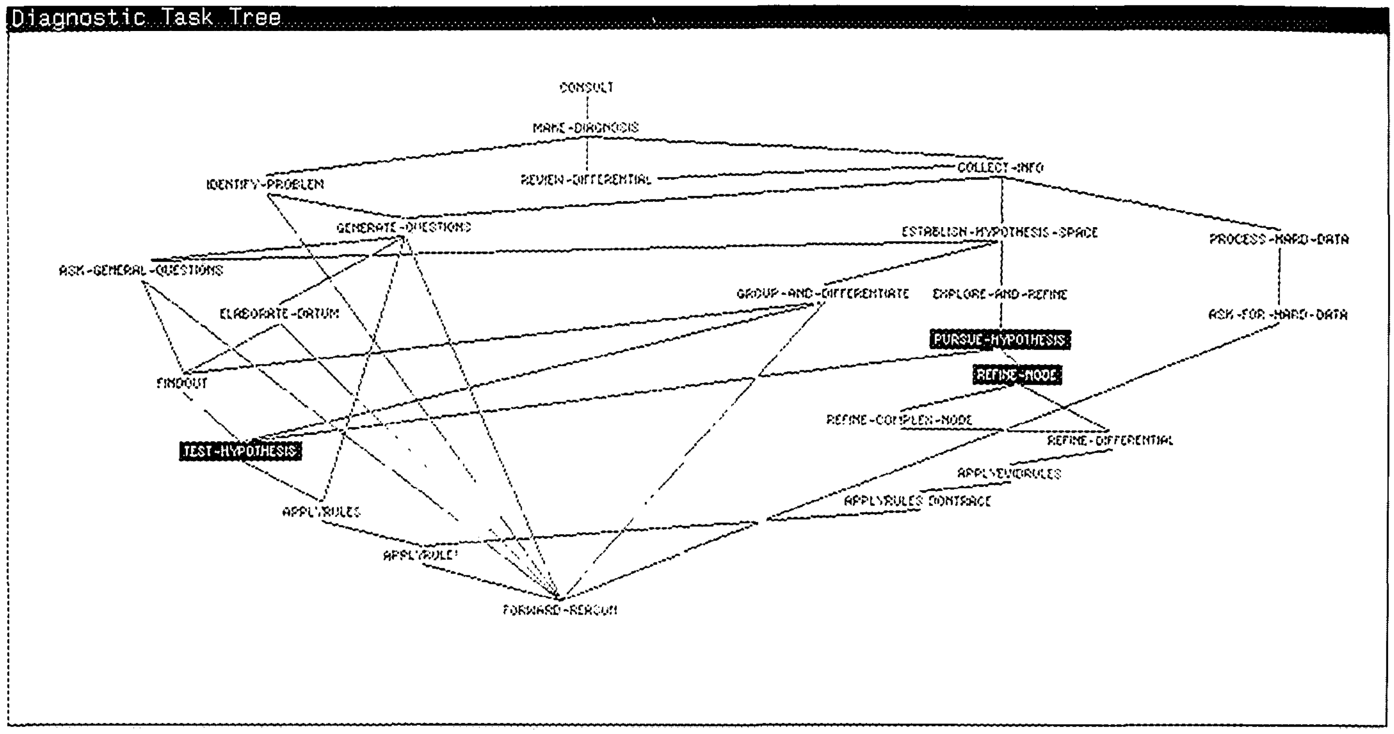


Figure 12. Invocation of Diagnostic Tasks, Shown as a Lattice. Each task is represented in NEOMYCIN as a stylized procedure, shown here as a node, with the subprocedures it calls below it. For example, pursuing a hypothesis involves testing and refining the hypothesis. To relate new findings and hypotheses, all tasks eventually call FORWARD-REASON, which invokes additional tasks not shown here GENERATE-QUESTIONS is invoked when there is insufficient information to proceed; it chases down leads in different ways, thus explaining its central position. Note also that FINDOUT calls TEST-HYPOTHESIS so that domain rules will be selected deliberately, replacing the back chaining of EMYCIN Using this representation for explanation and student modeling requires additional knowledge about task preconditions and postconditions and how metarules controlling task invocation are ordered.

Clancey 1984). Evaluation of these alternative approaches is in progress.

Figure 14 shows a parse of reasoning produced by the ODYSSEUS program. We're testing this program with "synthetic" students, systematically varying NEOMYCIN and comparing ODYSSEUS's interpretation to the known variations in the knowledge base. Another application is to give ODYSSEUS a sequence of data requests and to have it determine what knowledge base changes would be required to produce this sequence, consistent with the inference procedure. We believe that the simple classification nature of the inference procedure makes this approach plausible. We're developing this capability for a tutoring program called GUIDON-DEBUG (Clancey, *et al.* 1986). The same program could be used for knowledge acquisition.

Neoxpl: Strategic Explanation

Using NEOMYCIN's well-structured representation, Diane Hasling, Glenn Rennels, and I (1983) reformulated MYCIN's WHY/HOW explanations in terms of metarules and tasks. Figure 15 shows how procedural information is available prosaically (by asking WHY) or through the task stack.

Although our WHY/HOW system goes up the goal stack in a way similar to MYCIN's explanation program, this new program takes advantage of the structured representation to be more selective about what it says. In particular, it looks at the focus of a task to determine whether to mention the task as it goes up the stack. A focus can be one of three basic terms—a finding, a hypothesis, or a domain rule—or a list of these. If the focus is a rule or list of rules, the explanation program skips over the task (for example, APPLYRULES). The task is mentioned if its metarule establishes a new focus, such as going from a list of hypotheses to a single hypothesis (GROUP-AND-DIFFERENTIATE) or from a hypothesis to a rule (TEST-HYPOTHESIS). This turns out to be a good explanation heuristic. A new explanation system under development uses the propositional encoding of the metarules (described later) to select particular rule-premise clauses to mention.

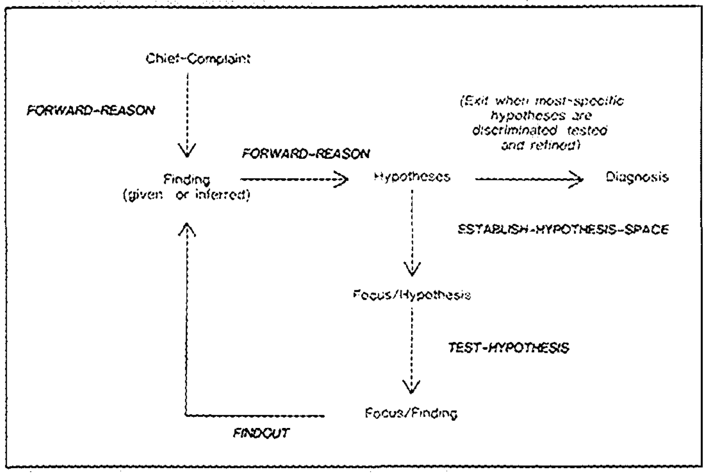

Chief-Complaint
FORWARD-REASON
FORWARD-REASON
(Exit when most-specific hypotheses are discriminated tested and refined)
Finding (given or inferred)
Hypotheses
Diagnosis
ESTABLISH-HYPOTHESIS-SPACE
Focus/Hypothesis
TEST-HYPOTHESIS
Focus/Finding
FINDOUT

Figure 13. Predominant Focus Shifts in Diagnosis. This diagram simplifies the dynamic flow of control between tasks, revealing how findings and hypotheses are related. New findings suggest new hypotheses and support existing hypotheses (FORWARD-REASON); a decision is made to focus on a particular HYPOTHESIS (ESTABLISH-HYPOTHESIS-SPACE); a decision is made to focus on a particular finding (TEST-HYPOTHESIS); the implications of the new information are considered, and so on. In contrast, MYCIN does not change its goals on the basis of new data or deliberately order the goals and data it will pursue. Accomplishing this by abstract metarules (not specifying domain terms) requires explicitly representing relations between findings and hypotheses, on the basis of which they will be selectively considered.

Tasks (appearing in bold italics) can be related to Figure 12, which shows the subtasks they invoke. In practice, ESTABLISH-HYPOTHESIS-SPACE is only invoked if there is reason to stop pursuing the current HYPOTHESIS. Criteria for applying domain rules in FORWARD-REASON are complex. For example, new findings are related to hypotheses "in focus"; if a new HYPOTHESIS "explains" the known findings at least as well as existing hypotheses, it is considered; new hypotheses are related to previously known findings, etc. The program stops when its differential, the list of most-specific hypotheses under consideration, has been discriminated, tested, and refined.

MRS/Neomycin: From Findings and Hypotheses to Relations

Student modeling, debugging, and explanation require that our programs reason about the premises of metarules, particularly to determine which domain facts matched and why rules failed. Originally, metarule premises were encoded in LISP. In a hybrid system called MRS/NEOMYCIN, Conrad Bock and I rerepresented metarule premises in MRS, a logic-programming language that provides a framework for multiple representations of knowledge and control of reasoning (Genesereth & Smith 1982). Bock also recoded the interpreter in MRS rules, and placed a simple deliberation-action loop at the top (Clancey & Bock 1982). Unfortunately, recoding the interpreter slowed down the program by an order of magnitude and made the procedure too obscure to read or maintain. In the current version of the program, we retain the original interpreter and use a variant of MRS as a specification language for metarule premises, which are compiled into Lisp. This provides the well-structured, uniform language our modeling and explanation programs require without sacrificing runtime efficiency. Figure 16 illustrates how MRS is used in the metarules and definitional rules for relations.

Primitive relations are compiled as direct LISP operations, using explicit declarations about how propositions are represented in the LISP-encoded knowledge base. For example, a TYPE proposition is represented as a property list structure, so the compiler substitutes a GETPROP, an ASSOC, or more complex loop construction, depending on what terms are known when the proposition is encountered in the metarule. In encoding propositions in standardized LISP structures, distinguishing between the language for expressing knowledge and how it is stored in the computer, we are exploiting the multiple representation aspect of MRS, which is one interpretation of its name. A number of elegant patterns in the metarules made the compiler easy to write (Clancey, forthcoming). Figure 17 summarizes how rules, tasks, and relations are encoded as EMYCIN rules and parameters and how these entities are related. Our success in building HERACLES on top of EMYCIN demonstrates the generality of the original parameter-rule representation language. It is closer to a typical frame language than is commonly realized.

The most exciting result of this reformulation is what it reveals about the relational nature of the knowledge base. It is now evident that the metarules are selecting foci (findings, hypotheses, domain rules) on the basis of how they are related to one another. These relations can be either static (for example, red-flag finding, one that needs to be explained) or dynamic (for example, hypothesis in focus). The knowledge base can be viewed as a database, defined in terms of these three primitive terms and relations among them. Writing a new metarule tends to require defining a new preference relation for discriminating among findings, hypotheses, and domain rules. That is, each new relation further classifies the primitive terms in a way useful for controlling reasoning. For example, the metarule shown in Figure 16 required the new relation, "a finding that needs to be explained." As this example shows, the meaning of a relation is tied to how the relation is used. This is particularly clear for relations such as follow-up question and trigger rule.

A detailed analysis shows that the metarules are collecting, sorting, and filtering domain terms and rules on the basis of their applicability as operands (foci) for the operators (subtasks) that will accomplish the current task.

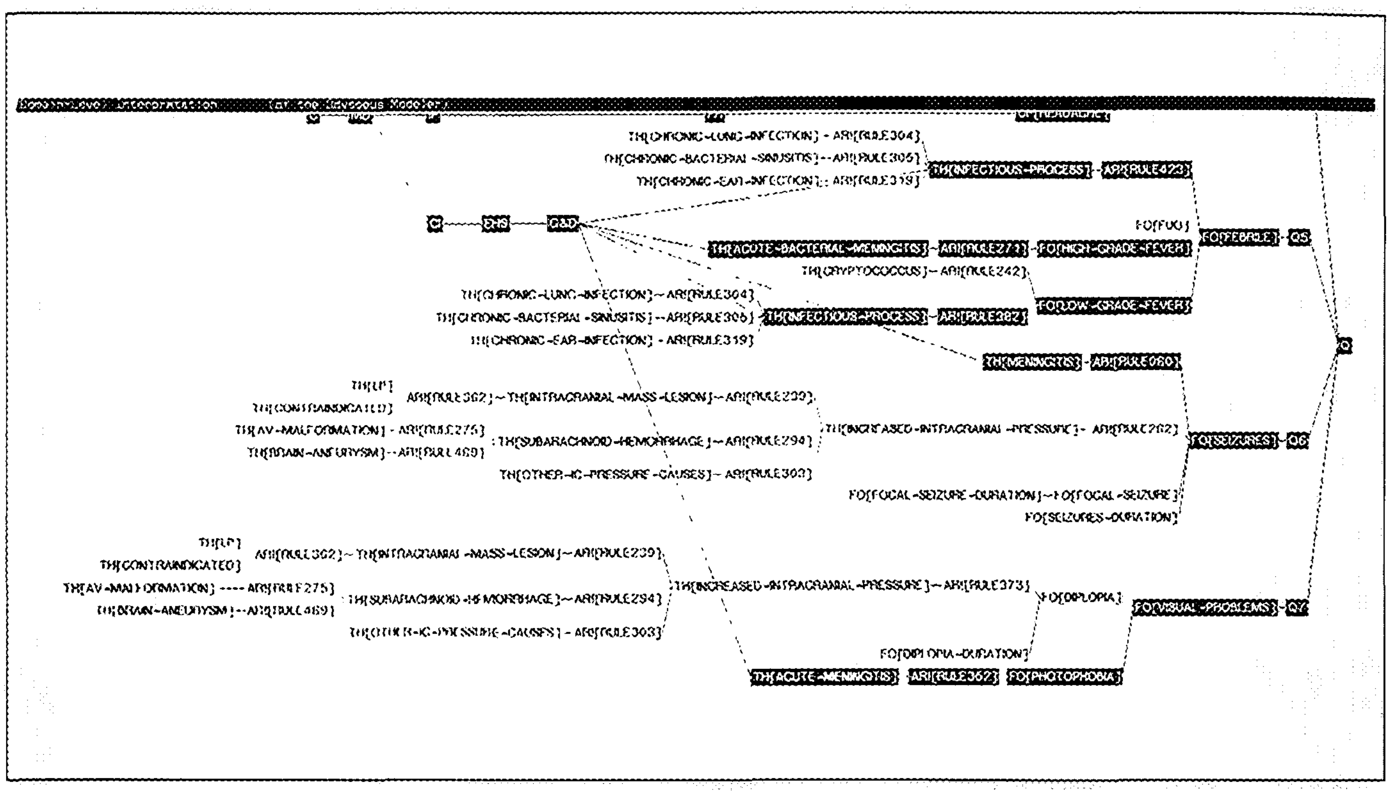

CI
EHS
G&D
TH{CHRONIC-LUNG-INFECTION} - ARI{RULE304}
TH{CHRONIC-BACTERIAL-SINUSITIS}--ARI{RULE305}
TH{CHRONIC-EAR-INFECTION} - ARI{RULE319}
TH{INFECTIOUS-PROCESS} - ARI{RULE423}
FO{FUO}
TH{ACUTE-BACTERIAL-MENINGITIS}-ARI{RULE271}-FO{HIGH-GRADE-FEVER}
FO{FEBRILE}-Q5
TH{CRYPTOCOCCUS}-ARI{RULE242}
FO{LOW-GRADE-FEVER}
TH{CHRONIC-LUNG-INFECTION}-ARI{RULE304}
TH{CHRONIC-BACTERIAL-SINUSITIS}--ARI{RULE306}
TH{INFECTIOUS-PROCESS}-ARI{RULE282}
TH{CHRONIC-EAR-INFECTION} - ARI{RULE319}
TH{MENINGITIS}-ARI{RULE060}
Q
TH{LP}
TH{CONTRAINDICATED}
ARI{RULE362}-TH{INTRACRANIAL-MASS-LESION}-ARI{RULE239}
TH{AV-MALFORMATION} - ARI{RULE275}
TH{BRAIN-ANEURYSM}--ARI{RULE469}
TH{SUBARACHNOID-HEMORRHAGE}-ARI{RULE294}
TH{INCREASED-INTRACRANIAL-PRESSURE}- ARI{RULE262}
FO{SEIZURES}-Q6
TH{OTHER-IC-PRESSURE-CAUSES}-ARI{RULE303}
FO{FOCAL-SEIZURE-DURATION}-FO{FOCAL-SEIZURE}
FO{SEIZURES-DURATION}
TH{LP}
TH{CONTRAINDICATED}
ARI{RULE362}-TH{INTRACRANIAL-MASS-LESION}-ARI{RULE239}
TH{AV-MALFORMATION} ---- ARI{RULE275}
TH{BRAIN-ANEURYSM}--ARI{RULE469}
TH{SUBARACHNOID-HEMORRHAGE}-ARI{RULE294}
TH{INCREASED-INTRACRANIAL-PRESSURE}-ARI{RULE373}
FO{DIPLOPIA}
FO{VISUAL-PROBLEMS}-Q7
TH{OTHER-IC-PRESSURE-CAUSES} - ARI{RULE303}
FO{DIPLOPIA-DURATION}
TH{ACUTE-MENINGITIS}
ARI{RULE362}
FO{PHOTOPHOBIA}

Figure 14. ODYSSEUS's Parse of a Student's Data Requests. Given a sequence of requests for patient data, listed on the right side of the figure (Q5, Q6, Q7), the program indicates all of the alternative justifications for why a question might have been asked. For example, the student's query about seizures (FINDOUT/Seizures, Q6) might have been asked to determine whether the disease is caused by an Intracranial Mass Lesion, Subarachnoid Hemorrhage, and so on. The program indicates in inverse video, combining its bottom-up analysis with a top-down parse, that this question relates to meningitis (TEST-HYPOTHESIS/Meningitis), as part of the process of discriminating hypotheses (GROUP-AND-DIFFERENTIATE). Thus, the problem state (hypotheses under consideration) and the tasks interact to explain finding requests in terms of a logic for focusing on hypotheses and findings. Note that by the same analysis the question about a fever (Febrile, Q5) has three consistent interpretations. This kind of analysis is not possible using MYCIN because, first, its reasoning did not involve "looking up" from "triggered hypotheses" and, second, its inference procedure is represented behaviorally as specific productions. A functional representation, as diagnostic tasks, relates surface behavior to abstract goals, which can be accomplished in multiple ways.

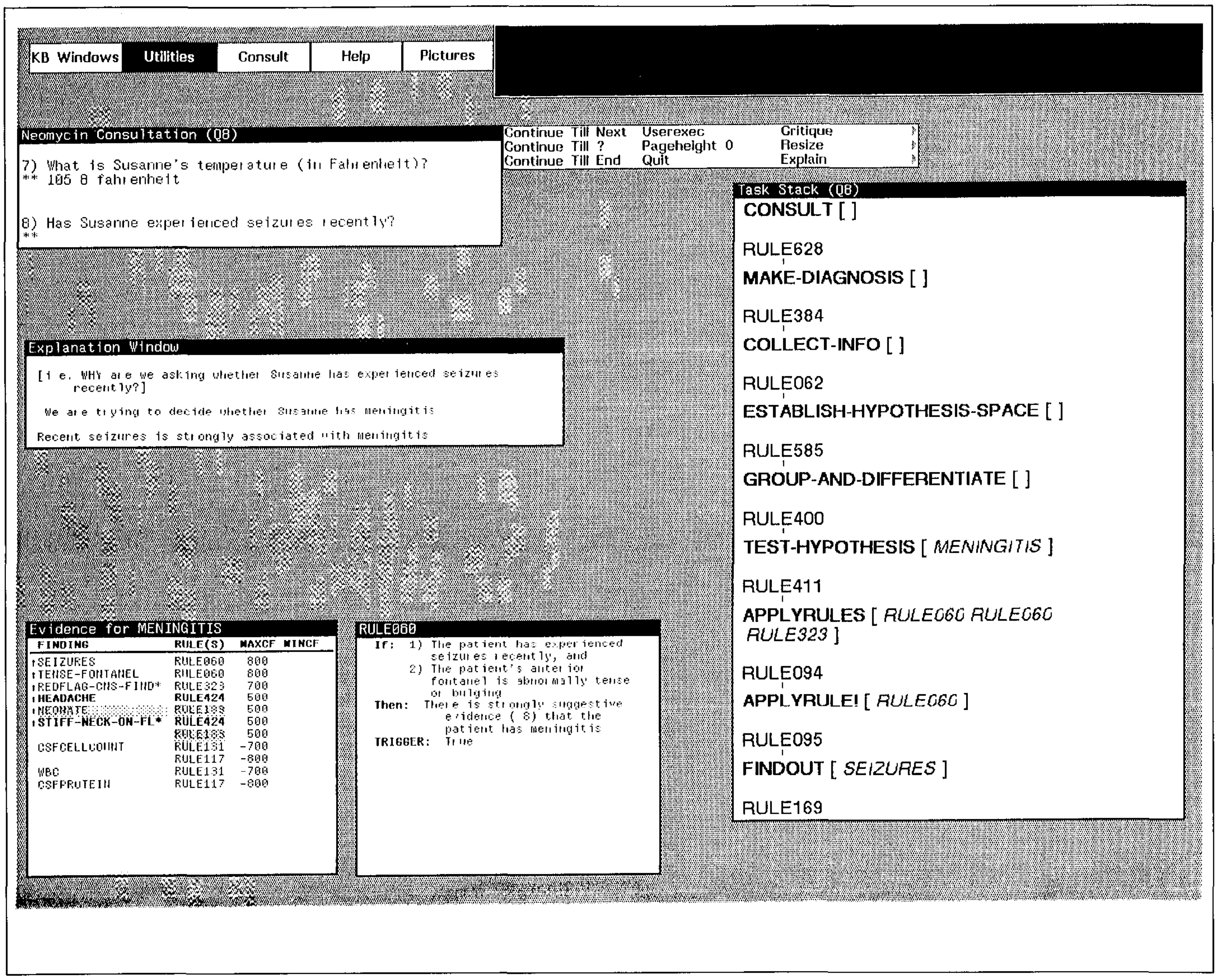
For example, metarules for TEST-HYPOTHESIS collect, sort, and filter potential findings to support a hypothesis. Refining a hypothesis means collecting, sorting, and filtering its causes and subtypes (for example, distinguishing between common and serious causes). Generally, the domain relations classify NEOMYCIN's experiential knowledge of predefined disease models (see Heracles: From Diseases to Stereotypes) according to how they are triggered, tested, discriminated, and refined by operators (tasks) for constructing a problem-specific, historical accounting of the disease process (see The Situation Specific Model: From a Diagnosis to an Explanation).

NEOMYCIN has about 170 relations in its control vocabulary. They appear in the 75 metarules, grouped into 29 tasks. In HERACLES, the generalization of NEOMYCIN, the knowledge engineer can modify these metarules, defining new relations for describing his domain.

Guidon-Watch: Reifying the Process

The availability of graphics has changed how we can illustrate reasoning and is shaping our ideas of what we'd like to show. As a first step toward implementing a new instructional program on top of NEOMYCIN, Mark Richer and I (1985) used the Interlisp-D window and menu features to construct a complex interactive system for browsing the knowledge base and watching reasoning. This includes the dynamic task tree (similar to Figure 14) and the task stack (see Figure 15). Our work has been directly inspired by Brown's emphasis on reifying or making concrete the reasoning process (Brown 1983).

Figure 18 shows how the disease taxonomy is overprinted to reveal the pattern of NEOMYCIN's reasoning. In GUIDON-DEBUG, now under development, it is possible to roll back the consultation display to show any window at the time any given question was asked. This is a debugging facility we could hardly have imagined even five years ago.



```
Neomycin Consultation (Q8)

7) What is Susanne's temperature (in Fahrenheit)?
** 105 8 fahrenheit


8) Has Susanne experienced seizures recently?
**
```

```
Explanation Window

[i e. WHY are we asking whether Susanne has experienced seizures
    recently?]

 We are trying to decide whether Susanne has meningitis

Recent seizures is strongly associated with meningitis
```

```
Evidence for MENINGITIS
 FINDING              RULE(S)   MAXCF  MINCF
↑SEIZURES             RULE060    800
↑TENSE-FONTANEL       RULE060    800
↑REDFLAG-CNS-FIND*    RULE323    700
↑HEADACHE             RULE424    500
↑NEONATE              RULE198    500
↑STIFF-NECK-ON-FL*    RULE424    500
                      RULE165    500
 CSFCELLCOUNT         RULE131   -700
                      RULE117   -800
 WBC                  RULE131   -700
 CSFPRUTEIN           RULE117   -800
```

```
RULE060
  If:   1) The patient has experienced
           seizures recently, and
        2) The patient's anterior
           fontanel is abnormally tense
           or bulging
  Then:  There is strongly suggestive
            evidence (.8) that the
            patient has meningitis
  TRIGGER:  True
```

```
Task Stack (Q8)
CONSULT [ ]

RULE628
MAKE-DIAGNOSIS [ ]

RULE384
COLLECT-INFO [ ]

RULE062
ESTABLISH-HYPOTHESIS-SPACE [ ]

RULE585
GROUP-AND-DIFFERENTIATE [ ]

RULE400
TEST-HYPOTHESIS [ MENINGITIS ]

RULE411
APPLYRULES [ RULE060 RULE060
 RULE323 ]

RULE094
APPLYRULE! [ RULE060 ]

RULE095
FINDOUT [ SEIZURES ]

RULE169
```

Figure 15. Multiple Views of the Diagnostic Process: Question, Evidence Relation, Task Stack, Metarules, and Prosaic Condensation. When NEOMYCIN asked about seizures (question 8), the user selected a subitem in the KB WINDOWS menu, which caused the *task stack*—the current line of reasoning—to be displayed. The rule above a task is the metarule that invoked it; thus, rule 400 selected meningitis as a focus, invoking TEST-HYPOTHESIS with it as an argument. Selecting meningitis in this window caused the table in the lower left to be displayed. Here, boldface type indicates positive findings and successfully applied rules. Greyed areas correspond to negative findings and failed rules. Thus, the patient is not a neonate; rule 424 succeeded. Arrows preceding a finding indicate that the finding is in a triggering relation with the hypothesis. For example, the headache volunteered in the chief complaint caused the program to try to apply rule 424. When the user selected EXPLAIN in the menu adjacent to the consultation typescript, the program summarized the line of reasoning, skipping over "uninteresting" tasks

Heracles: From Diseases to Stereotypes

In late 1983 I began to consider how NEOMYCIN might be generalized. What kinds of problems can be conveniently solved by an architecture consisting of a classification network and a separate, abstract control strategy? In particular, to what problems can the same diagnostic strategy be applied? It was obvious from the start that the procedure had nothing specifically to do with medicine; was it more general than diagnosis? In attempting to teach the NEOMYCIN approach to AI students, I found that it was possible to redescribe other knowledge bases in its terms. For example, in terms of the mapping between models of situation descriptions and selected solutions, "people are to diseases" as "meals are to wines." I had also recently reread Rich's work on user modeling (Rich 1979), intending to apply this to our explanation program. I recognized that it fit the same pattern—models of people related to a taxonomy of books. Finally, I recalled that Rubin (1975) and Aikins (1983) emphasized that diseases are described

```
TASK: PROCESS-FINDING
FOCUS: $FINDING

Metarule

IF:     (AND (OR (FINDINGTYPE $FINDING REDFLAG)
                 (NOT (DIFF.EXPLAINED $FINDING)))
             (MAKESET (TRIGGERS? $FINDING $RULE)
                      RULELST))
THEN:(TASK APPLYRULES RULELST)

If the finding must always be explained or
it is not currently explained by the differential,
then trigger hypotheses that explain it.

Definitional relation rules

IF:     (AND (DIFFERENTIAL $HYP)
             (EXPLAINEDBY $FINDING $HYP))
THEN:(DIFF.EXPLAINED $FINDING)

A finding is explained by the differential
if it's explained by some hypothesis in the differential.

IF:     (OR (CAUSED-BY $F $H)
            (AND (TYPE $H $PARENT)
                 (EXPLAINEDBY $F $PARENT)))
THEN:(EXPLAINEDBY $F $H)

A finding is explained by a hypothesis
if it is caused by the hypothesis or
by some more general category.
```

Figure 16: Propositional Representation of a Metarule. This is one of six metarules for accomplishing the task PROCESS-FINDING, which is invoked whenever a new finding becomes known. The metarule detects that this finding is serious and has to be explained (a red-flag finding), or it's something that's not currently explained by the set of possibilities under consideration. The program gathers up the *trigger rules*—automatic inferences—and tries to apply them. The idea is that if the finding doesn't always have to be explained and it's explained by hypotheses that were already triggered, you shouldn't trigger a new hypothesis. For example, if the patient has a headache, and other evidence suggests meningitis, which would explain the headache, there's no need to consider other explanations of the headache. Intermediate relations, such as EXPLAINEDBY, are defined by other rules (simplified here). All pattern variables in these rules are instantiated as domain rules or terms. All expressions are implicitly universally quantified.

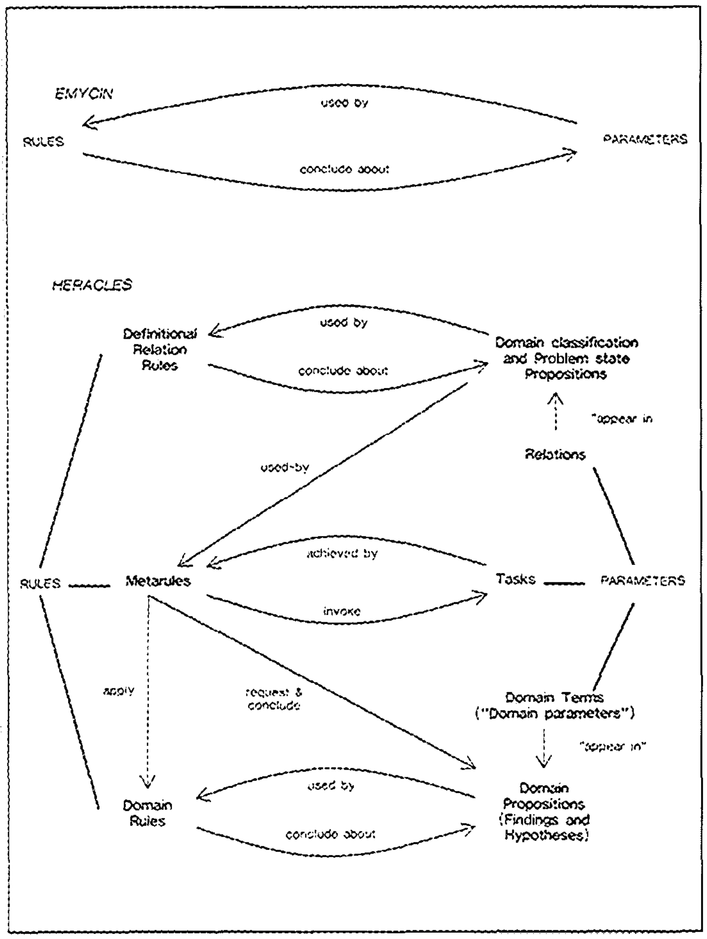
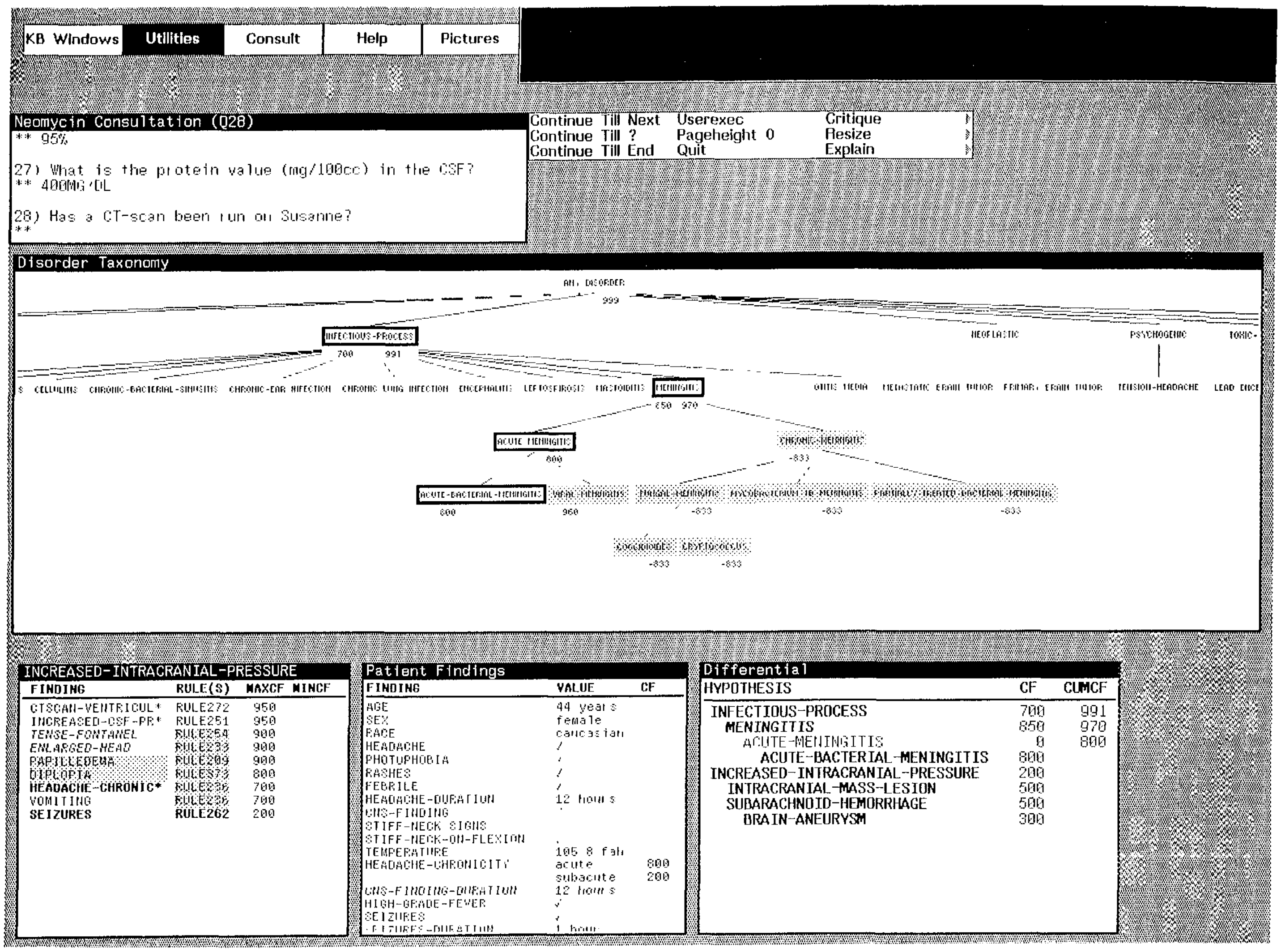


Figure 17. How Control Knowledge is Encoded in HERACLES. HERACLES is implemented as a specialization of EMYCIN. Above is shown the original conception of domain parameters and rules. In HERACLES parameters are specialized as domain relations, control tasks, and domain terms, conditionally inferred and invoked by rules. We use "relation" in the mathematical sense to refer to both predicates and functions. "Finding" and "hypothesis" formally classify the domain terms (for example, meningitis), and informally are used to refer to propositions in a situation-specific model; so, we say that "the patient has meningitis" is a hypothesis.
Tasks are accomplished by an interpreter that applies metarules. Propositions used by metarule premises (such as (EXPLAINED-BY $F $H) appearing in Figure 16), can be inferred definitionally by rules or can be inferred by procedural attachment (for example, accessing Lisp structures). These propositions are both static and dynamic. They classify domain propositions and domain rules, as well as characterizing the problem-solving state (such as whether a hypothesis is in the differential or whether a task has been done yet). Additional relations that classify tasks are used by the task interpreter (not shown here). Metarule actions apply domain rules, request (from the user) or conclude domain propositions, or invoke other tasks. In particular, the task FINDOUT uses all of these methods to infer domain propositions. In HERACLES all domain rules are applied directly by metarules rather than by back chaining. Only domain rules mention domain terms directly; other rules use variables.



```
Neomycin Consultation (Q28)
** 95%

27) What is the protein value (mg/100cc) in the CSF?
** 400MG/DL

28) Has a CT-scan been run on Susanne?
**
```

INCREASED-INTRACRANIAL-PRESSURE

FINDING	RULE(S)	MAXCF	MINCF
CTSCAN-VENTRICUL*	RULE272	950	
INCREASED-CSF-PR*	RULE251	950	
TENSE-FONTANEL	RULE254	900	
ENLARGED-HEAD	RULE233	900	
PAPILLEDEMA	RULE209	900	
DIPLOPIA	RULE573	800	
HEADACHE-CHRONIC*	RULE236	700	
VOMITING	RULE236	700	
SEIZURES	**RULE262**	200	

Patient Findings

FINDING	VALUE	CF
AGE	44 years	
SEX	female	
RACE	caucasian	
HEADACHE	✓	
PHOTOPHOBIA	✓	
RASHES	✓	
FEBRILE	✓	
HEADACHE-DURATION	12 hours	
CNS-FINDING	'	
STIFF-NECK SIGNS		
STIFF-NECK-ON-FLEXION	.	
TEMPERATURE	105 8 fah	
HEADACHE-CHRONICITY	acute	800
	subacute	200
CNS-FINDING-DURATION	12 hours	
HIGH-GRADE-FEVER	✓	
SEIZURES	✓	
SEIZURES-DURATION	1 hour	

Differential

HYPOTHESIS	CF	CUMCF
INFECTIOUS-PROCESS	700	991
MENINGITIS	850	970
ACUTE-MENINGITIS	0	800
ACUTE-BACTERIAL-MENINGITIS	800	
INCREASED-INTRACRANIAL-PRESSURE	200	
INTRACRANIAL-MASS-LESION	500	
SUBARACHNOID-HEMORRHAGE	500	
BRAIN-ANEURYSM	300	

Figure 18. Overprinting a Classification Network to Show How It Is Searched. Nodes blink and are boxed to make visible the "looking up" and "looking down" process of diagnosis. Numbers indicate the relative certainty of conclusions; the cumulative certainty factor (CUMCF) includes hierarchical propagation. Heavy-bordered boxes indicate the program's *differential*—the most specific cut through the taxonomy and causal network. The differential is printed in the lower right window with indenting to show specialization by process subtype and cause. When a hypothesis is selected, the evidence window can be displayed, indicating which findings and rules have been considered and the outcome of each consideration. Dozens of other windows are available, including different views of causal networks and the history of task invocation.

(in knowledge bases) as stereotypes. The general model of heuristic classification fell into place: Some problems can be solved by selection, heuristically relating a classification of problem data to a classification of known solutions (Clancey 1985a).

To my chagrin, this new model required a reconceptualization of parts of NEOMYCIN. We began to consider the diseases as stereotypes, we introduced qualitative abstraction of numeric data where it had been omitted in MYCIN, and we realized that our representation of diseases as classes is inadequate given what is required in general and what is evident in other programs (for example, allowing for multiple inheritance). We call the reconceptualized framework HERACLES. It is not a completed tool but an idea that continues to evolve.

Figure 19 illustrates the heuristic classification analysis of SACON, a program that many of us knew about and talked about for five or six years, but that few understood until its knowledge base was portrayed in this way. The purpose of SACON is to select a configuration of programs in a structural analysis software package developed by the Marc Corporation. These programs can analyze an object for structural failure in many ways, some of which are unnecessarily accurate and time consuming. An expert can tell you which of the programs should be run to analyze a particular structure, and that is SACON's task. Imposing a type classification on SACON's concepts, and labeling inferences as abstractions, heuristics, and refinements, we find a previously hidden secondary structure which helps us to understand what SACON does.

Studying and generalizing knowledge-based programs, we can go quite a bit further. First, we can realize that as stereotypes the classifications are models of systems: specifications or descriptions of systems and plans for assembly or modification of systems. Second, the classification sequences, relating one model to another, are regular and limited in nature, constituting tasks. A model of system being monitored is related to a plan for controlling its behavior. A diagnostic model of a faulty system is related to a repair plan. A specification is related to a design and then to an assembly plan. Finally, the idea of systems, tasks, and common sequences is independent of how the solutions are computed each step along the way. Either heuristic classification or some constructive method (perhaps involving nonmonotonic reasoning, hypothetical worlds, and so on) might be used. It is important to remember that this inference structure shows the pattern of inferences that map given information to final solutions, and makes no claims about the process or order in which the inferences are made. Further examples and extensive discussion appear in (Clancey 1985a, 1986).

Caster: From Diseases to Abnormal Substances and Processes

In addition to the disorder taxonomy (Figure 11), a knowledge base for diagnostic problems constructed in HERACLES might include a causal-associational network. Disorders in this network are descriptions of internal states in the system being diagnosed. Figure 20 shows such a network for CASTER, a knowledge system for sand-casting diagnosis.

Tim Thompson and I (1986) developed this program in order to better understand the distinction between the pathophysiological states of the causal net and the *etiologies*, or final causes, of the disorder taxonomy. This distinction was emphasized in the CASNET program (Weiss *et al.* 1978); our interest was to apply the ideas to a nonmedical problem.

What did we learn from the CASTER experiment? First, for diagnosing malfunctions in some manufacturing process, it is useful to organize the disorder taxonomy according to each stage in the overall process (pattern design, melting, and so on). In contrast, the top level of NEOMYCIN's taxonomy corresponds to defects in the neurological system, viewing it as an object, not a process: assembly flaw (congenital), environmental influence (infection, toxicity, trauma, psychological load), or degeneration (vascular disorder, immunoresponse, muscular disorder). In both of these physical systems, externally observable manifestations are explained in terms of internal system behavior, tracked back to faulty structures and malfunctions of subsystems. These are in turn explained by the etiologies, processes in which the system interacted with its environment, bringing it to its current state. In medicine, these etiologies include congenital problems (caused by the mother's lifestyle or her environment), psychogenic problems (emotional overload), trauma (structurally damaging the body), toxic environment, and so on. In the human body, internal systems generate new subsystem structures, so developmental and degenerative processes are also important etiologies. We believe that this analysis can be generalized to cover all physical systems.

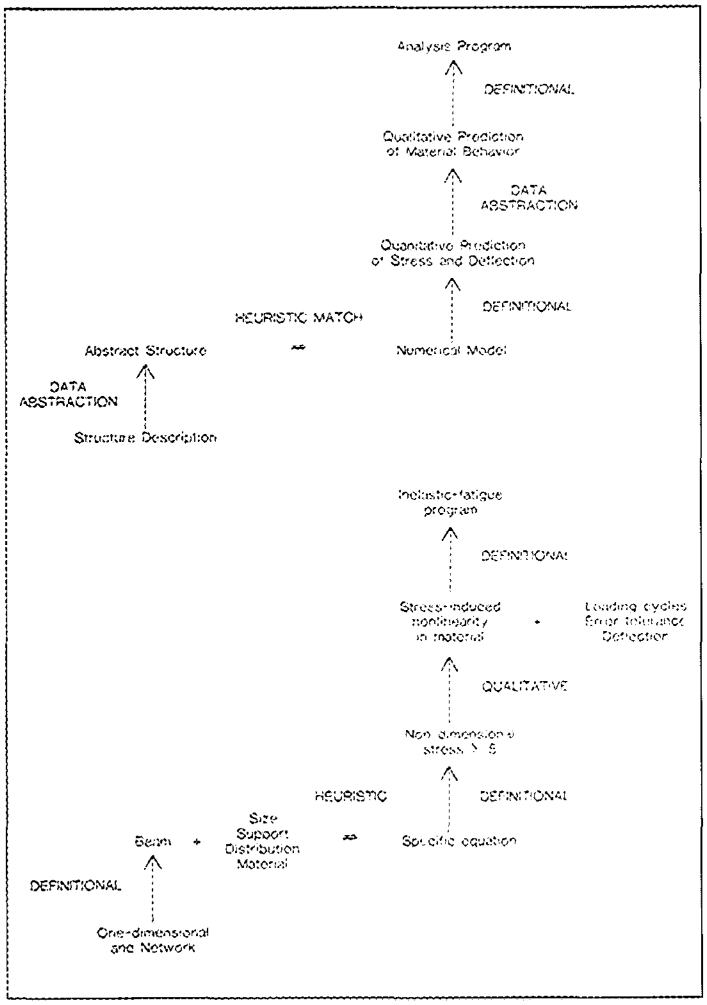
A second interesting result is the set of heuristics we discovered for constructing a well-formed causal network (Clancey 1984d). These heuristics include asking the expert about categories of states; asking about unobservable states that track back to different etiologies; distinguishing clearly between substances and processes, particularly, never causally linking substances directly; and working backward from repairs to causes. This last point emphasizes that the purpose of the causal-associational and etiologic taxonomy is to make choices about repair, a point I emphasized in The Disease Taxonomy: Searching an Abnormal Process Clarification. Uncertainty in diagnostic reasoning need only be resolved to the extent that it makes a difference in distinguishing among repairs.

Our heuristics can be viewed as criteria for critiquing a behavioral causal model. Can we formalize these constraints so that they can be taught to a student? Viewing a diagnosis as a model is the first step.

The Situation-Specific Model: From a Diagnosis to an Explanation

This lesson might be the most important. It is the idea that a diagnosis is not the name of a disease but an *argument* which causally relates the manifestations which need to be explained (because they are abnormal) to the processes that brought them about (See Figure 21). A number of ideas come together here:

- Diseases are processes (see The Disease Taxonomy: Searching an Abnormal Process Classification and Caster: From Diseases to Abnormal Substances and Processes.) Thus, a diagnosis is a network causally linking manifestations and states to processes.
- A causal explanation applies the general concepts and links in a knowledge base to construct a case-specific model (Patil, Szolovits, & Schwartz 1981). Thus, the network linking manifestations and diseases is a model of a particular sequence of events in the world (also called a *situation-specific* model).



Analysis Program
DEFINITIONAL
Qualitative Prediction of Material Behavior
DATA ABSTRACTION
Quantitative Prediction of Stress and Deflection
DEFINITIONAL
HEURISTIC MATCH
Abstract Structure
Numerical Model
DATA ABSTRACTION
Structure Description
Inelastic-fatigue program
DEFINITIONAL
Stress-induced nonlinearity in material
+
Loading cycles Error tolerance Deflection
QUALITATIVE
Non-dimensional stress > .5
HEURISTIC
DEFINITIONAL
Beam
+
Size Support Distribution Material
Specific equation
DEFINITIONAL
One-dimensional and Network

Figure 19. Inference Structure of SACON. An abstract description of inference chains is shown above a particular sequence of associations. SACON abstracts the given structure and relates this abstraction to half a dozen rules of thumb that make a quantitative prediction of the structure's behavior under stress. Specifically, the fact that the structure is a beam is combined with information about its size, support, and load distribution in order to select a numeric equation, which computes stress and deflection. These predictions are abstracted, and definitionally related to the structural analysis program. Specifically, the characterization of stress, combined with information about loading and error tolerance, is classified as a particular kind of "analysis" for which the program is specialized. The SACON program selects from about 30 different program combinations. This corresponds to the number of organisms in MYCIN, and is probably good to remember when considering whether the heuristic classification method is appropriate for solving a problem.

- Diagnostic operators examine and modify the differential (most specific diseases under consideration), linking and refining them. Thus, HERACLES tasks are operators for constructing a situation-specific model (similar to ABEL's diagnostic operators [Patil, Szolovits, & Schwartz 1981]).
- A causal explanation has the structure of a geometry proof: It must account for all of the findings and must be coherent and consistent. Thus, the situation-specific model must be a connected graph with one process at the root (assuming a single fault).

The evolution of these ideas is intriguing, revealing how our computational tools and the use of the computer as a modeling medium changes how we think. Sometime in 1985 it occurred to me that we could extend the windows offered by GUIDON-WATCH to include a graph showing how the final diagnosis related to the known findings. When I saw the way Anderson replaced a linear geometry proof by a graph (using the same Interlisp-D graphics package), the analogy between a causal explanation and a proof became concrete (Anderson, Boyle, & Yost 1985). Thus, the example from another domain showed how Patil's idea of a patient-specific model could be useful in teaching, and the availability of the graphics package encouraged us to create the picture to see what it would look like.

It is astounding to realize how many hundreds of expert systems are cranking out diagnoses with neither the programs nor their designers ever explicitly considering a diagnosis as a coherent causal model. They don't even check to see if all of the findings are covered by the final diagnosis. Our language is too loose: The program prints out the name of a disorder, and we say, "The program has made a diagnosis." However, where is the explanation argument?

For the purpose of teaching, this graph could perhaps be the best way to reify the process of diagnosis. For several years, inspired by Brown's emphasis on "process versus product" (possibly derived from Dewey [1964]), I've been searching for some written notation that we could use, something analogous to algebra, to make visible what the operators of diagnosis (NEOMYCIN's tasks) 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 all axioms.

Giving this window to the student, we might have him carry out the diagnosis by posting his hypotheses and linking them to the known findings. Each step along the way, there are visible problems to be solved. The student can see that he is trying to construct a logically consistent network. Behind each request for data is an operation for making the network hang together—explaining the findings that need to be explained and refining the hypotheses that need to be made more specific. An instructional program is now being developed based on this idea. Called GUIDON-MANAGE, it has a student "manage" the diagnosis by explicitly applying strategic operators.

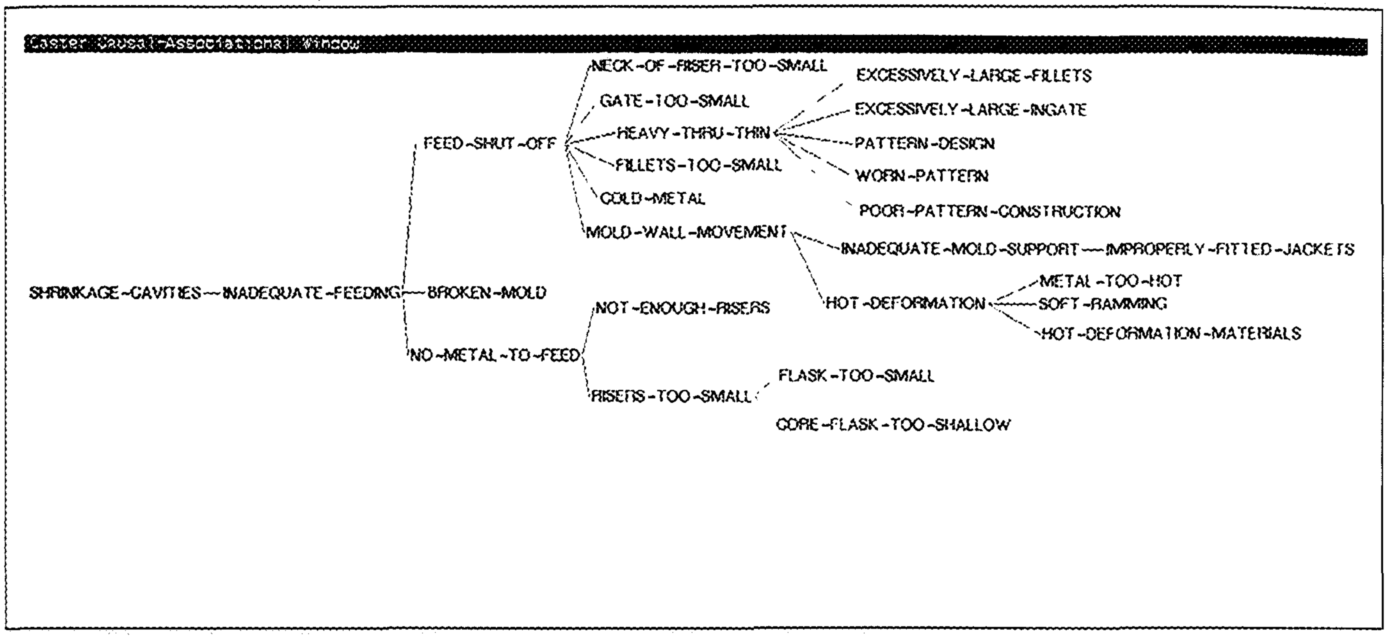


Figure 20. CASTER's Causal-Associational Network for Shrinking Defects in Cast Iron. This simplified network relates structural failures (for example, mold wall movement) to functional failures (for example, inadequate mold support). These are all internal to the system and often can't be observed directly Reasoning proceeds as follows. Given some surface fault, such as shrinkage cavities in the cast iron, we reason backwards to possible causes: (1) feed of metal shut off, (2) a broken mold (leak), and (3) absence of metal to feed. Gates, risers, and fillets refer to structures for shunting metal and venting gases. Terminal nodes, on the right side, track the problem back to some problem in the iron-casting process (pattern design, mold formation, metal melting, and so on), thus relating system behaviors to external causes (the designer's assumptions, previous treatment of the sand, contamination of the metal supply, and so on). We believe that analyzing such networks, relating them to the well-defined structure and function of the sand-casting system, will help us to redefine in a principled way the causal relations given to us by experts in other domains, such as medicine. Working in multiple domains proliferates metaphors and helps us to develop more general theories about expert knowledge.

This is an amazing change. Ten years ago I thought I was trying to teach parameters and rules, and now I'm saying that I want to teach the student to be an efficient model builder. What can we tell the student that will help him critique the model that he's constructing? For example, we'll say, "All the important findings need to be explained." Observing that he has failed to do something that needs to be done, we'll tell him about the operators, so he can step back and say, "Well, what knowledge might I be missing that prevented me from carrying out that task?" So debugging by explanation of failure—proceeding from model constraints to operators to knowledge relations—is the approach we're following. This leads to an interesting model of learning (Clancey *et al.* 1986).

Methodological Lessons

To summarize ongoing projects mentioned or alluded to here, we are currently doing the following:

- Developing instructional programs based on NEOMYCIN
- Studying learning in the setting of debugging a knowledge base
- Reimplementing the explanation program to use the logic encoding of the metarules (stating this program in the same task-metarule language so that it might reason about its own explanations)
- Generalizing our graphics package using object-oriented techniques
- Applying the student-modeling program, ODYSSEUS, to knowledge acquisition
- Preparing HERACLES for use by other people

I'm going to jump up a level here to consider some methodological lessons we can draw from this research.

Figure 1 provides a simplified summary of how the various programs and research ideas are connected. We observe two examples of a specific expert system being generalized, with the resulting shell used to construct other specific systems and a tutoring shell. Is there any logic in this sequence that might reveal something about learning in general or at least about how we learn by constructing programs?

In the section names in this article, I indicated the sequence of terminological changes ("from ... to ...") that seem to mark each major change in my understanding.

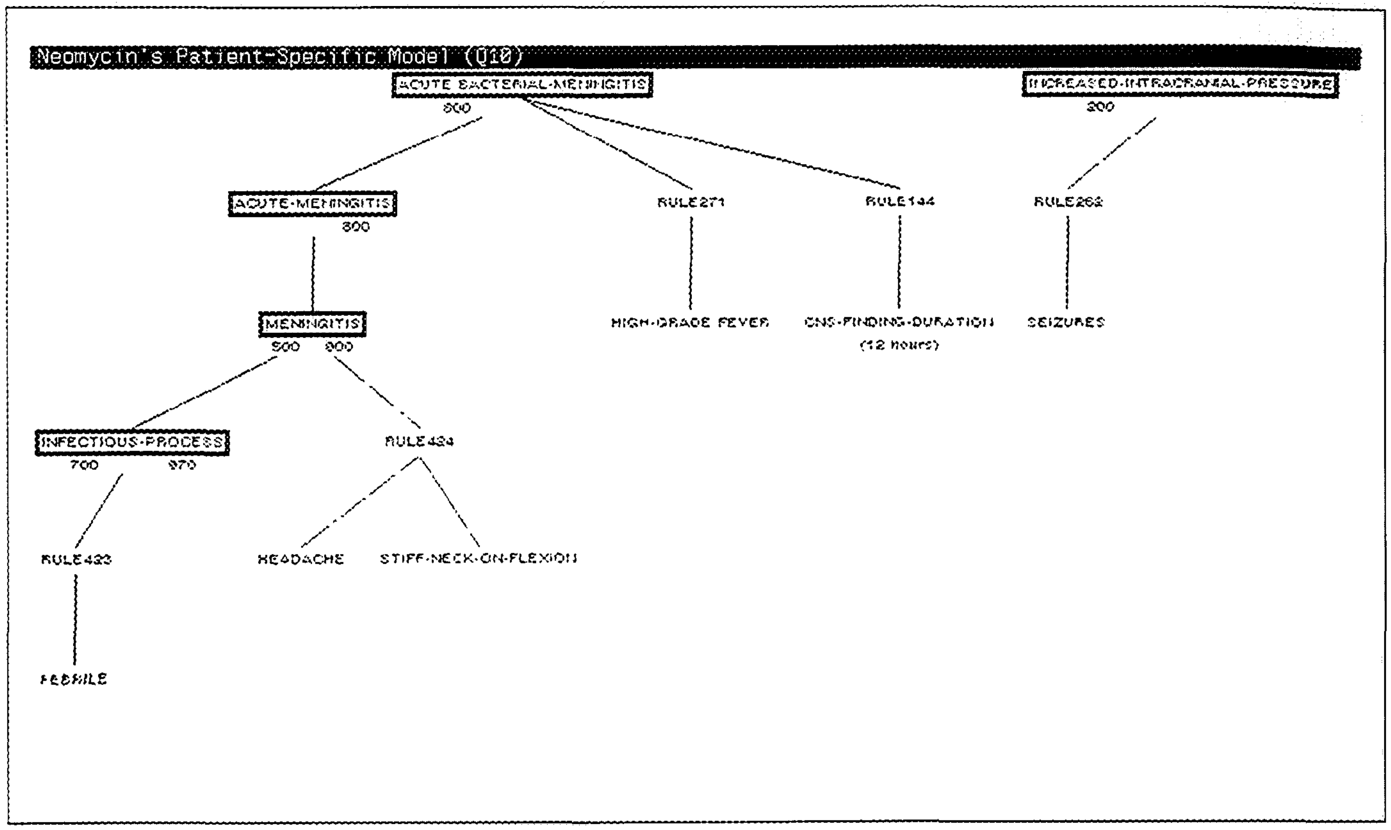


Figure 21. Partial Diagnostic Model in NEOMYCIN. The process of diagnosis is the construction of a *proof tree,* relating the findings and disorders that could have caused these findings. At some intermediate state when solving the problem, the network is disconnected and partial. The patient has seizures; what could have caused that? There is some support for Acute Bacterial Meningitis and Increased Intracranial Pressure, but these two hypotheses haven't been related. Is there some underlying cause (process) that could account for all of the manifestations? Diagnostic operators can be viewed as graph construction operators, focusing on particular nodes and trying to grow the graph down to support possible explanations or refining it upward to more specific explanations. A final situation-specific model is a connected network, with some root process that we say explains the internal states (such as Increased Intracranial Pressure), which, in turn, explain the observed findings. This graph, as an argument having the structure of a proof, is the diagnosis, not the term Acute Bacterial Meningitis.

The renaming that occurred in moving from "clinical parameter" to "model" is dramatic. None of the intermediate concepts (hypothesis, relation, process, and so on) is new, but it is interesting to note how they are retained and how they build upon one another as the knowledge structures are reinterpreted from different perspectives.

Thus, in HERACLES today, we have parameters, terms, hypotheses, diseases, processes, stereotypes, and models. All of these remain true descriptions of what's in our program. The perspective changes, broadening from *language terminology* (parameters, terms) to *reasoning phenomenology* (hypotheses), *domain ontology* (disease process taxonomy); and, finally, *epistemological distinctions* (stereotypes, models). With the heuristic classification perspective at the top—couched in terms of systems, tasks, and models (Clancey 1986)—previous terminology is retained for describing the program at different levels.

Looking closely at the sequence of research itself, there are some clear patterns:

- Abstracting or generalizing terminology to incorporate another specific domain (for example, moving from disease to disorder process)
- Separating a domain model (what is "true") from the inference process (what to do) by identifying and justifying procedural sequences (for example, defining relations for ordering MYCIN's rule clauses and, later, defining relations for ordering NEOMYCIN's metarules)
- Justifying domain relations in terms of underlying constraints and patterns (for example, a theory for generating appropriate follow-up questions or trigger rules or a theory for generating a causal network in terms of faulty structures and malfunctions)

Figure 22 summarizes the overall pattern. The point of the analysis phase is to detect patterns that we want to model explicitly and that have been mapped into the language in an implicit and perhaps undisciplined way. Thus, findings and hypotheses, causality and subtype, and disease knowledge and procedures are not distinguished in MYCIN. Findings and hypotheses are both represented as parameters. Cause and subtype are represented by sequences of clauses in rules, or in the relation between a parameter and its values (for example, parameter—"the kind of Meningitis"; value—"Bacterial"). Focusing procedures are also encoded by rule clause ordering.

There is apparently no end to this criticism; the same game can be played with NEOMYCIN. For example, in attempting to improve the explanation program we find that the use of terms in NEOMYCIN's original metarules is ludicrously undisciplined; they are used like arbitrary program variables, with no apparent connection between $HYP and $CURFOCUS. Interpreting this representation for diagnosis causes no difficulties, but the explanation program needs to know that the metarules refer to the same kind of entity, a hypothesis.

This analysis suggests that detecting patterns of statements in some language, articulating a new classification model, and defining a new procedure by which the statements are to be interpreted are intricately related. Recalling the analysis of metarules (MRS/NEOMYCIN: From Findings and Hypotheses to Relations), we observe that each new purpose for interpreting a representation requires new distinctions—new relations—to classify existing domain terms, rules, and relations among them. Thus, the compiler needs to know which domain relations are predicates and which are functions (in the mathematical sense). ODYSSEUS needs to know when metarules can be reordered. The teaching program needs to know why metarules are ordered a certain way. In classifying relations and terms, we are constantly asking, "Which things can be procedurally operated upon in the same way?"

Winograd reached the same conclusion in his analysis of how language arises. The need to take action reorients us to the world, forcing us to make new distinctions. The relevant properties attributed to an object are determined by the role the object plays in an action: "This grounding of description in action pervades all attempts to formalize the world into a linguistic structure of objects, properties, and events" (Winograd & Flores 1986). Indeed, by this analysis the world and its objects exist only in language, mediated by action.

The expert system methodology of writing down knowledge in some structured way so that it can later be studied and better formalized is a remarkable, exciting turning point in epistemological practice. We try to understand why a relation holds by abstracting it and then trying to find similar relations in the knowledge base. If a pattern holds, we restate everything more abstractly. Why is it correct to say that "broken mold" causes "inadequate feeding"? What other causal links in the network connect

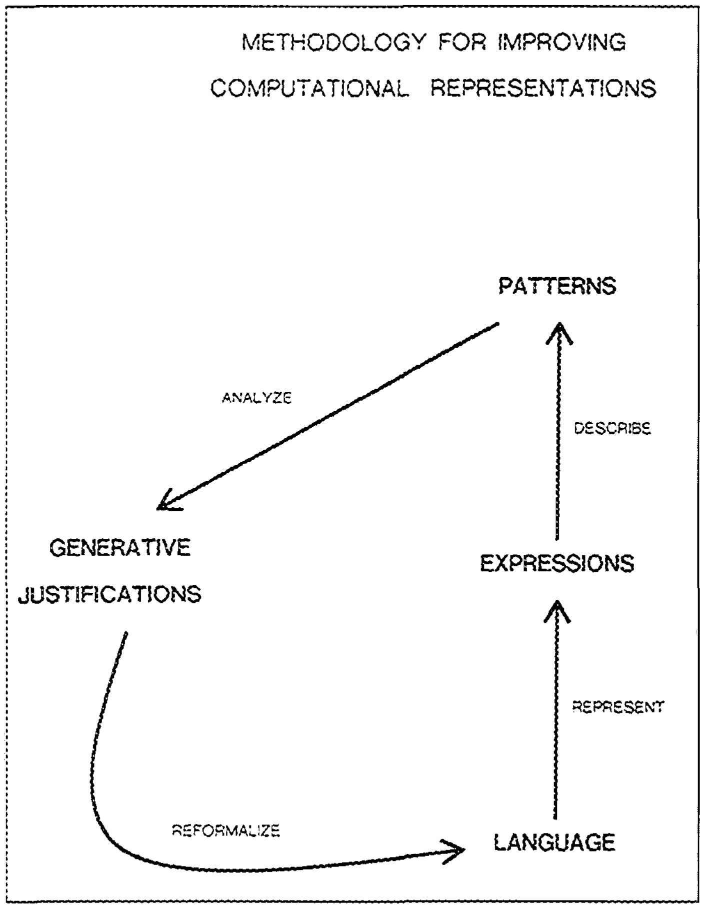

METHODOLOGY FOR IMPROVING
COMPUTATIONAL REPRESENTATIONS
PATTERNS
ANALYZE
DESCRIBE
GENERATIVE JUSTIFICATIONS
EXPRESSIONS
REPRESENT
REFORMALIZE
LANGUAGE

Figure 22. Methodology for Improving Computational Models. In the process of knowledge representation, we write statements in some language; we organize what we have written down, describing and classifying patterns; we explain the patterns in terms of primitive relations; and we define a new language that enables us to explicitly state these primitive relations and generate the original patterns. For example, clause correlations in in MYCIN's rules are now reformulated in tasks, metarules, and domain relations. Another cycle occurs when we study these metarules and articulate the constraints behind their design. Similarly, patterns in NEOMYCIN's disease taxonomy and CASTER's causal network are articulated by characterizing diseases as processes and states as abnormal structures and malfunctions. These new perspectives—the search for patterns and their articulation in a new language—all rise in an attempt to formulate some generative rationale for constructing similar structures in new domains as well as to evaluate existing networks for consistency and completeness. A generative theory of a representation facilitates teaching people how to use the representation, reformulating it for efficiency, and constructing explanation programs and knowledge-acquisition tools.

the same kind of concepts, leaving out the same kind of details? Do all links in the network connect structures to functions? Is there any reason why they should?

Having written a model down, the most powerful tools of language come into play:

- It is possible to reflect on what was said, to ask why it is true, to develop a better understanding or theory.
- An incremental critique and transformation process becomes possible—the best way to build anything so that it is reliable and useful (Petroski 1985).

Computational languages provide a way of writing things down so that the model is executable, the very thing we need for modeling processes. AI research is exploring how to model physical, inferential, communicative, motoric, and perceptual processes using qualitative (principally nonnumeric, relational) representations (Clancey 1986). Graphics provide a medium for visualizing processes, so we can understand the complexity of the systems we construct (Hollan, Hutchins, & Weitzman 1984; Clancey 1983c, Richer & Clancey 1985) and even start to ask new questions as icons and graphs become part of our language for stating theories. The marriage of qualitative modeling and graphics in the 1980s, made available on cheaper, more powerful machines, provides a sharp stimulus to AI research and a good reason to be optimistic about the progress to come.

Acknowledgements

This article was prepared as a final ONR technical report for the period 1979-1985 The GUIDON project was first funded jointly by Marshall Farr at ONR and Dexter Fletcher at DARPA Other tutoring programs in the ONR/ARPA family at that time included WHY, WEST, and SOPHIE. Avron Barr, John Seely Brown, Ira Goldstein, and Keith Wescourt helped us to define the initial proposal, conveying the problems and methods then current in the research community. Bruce Buchanan was principal investigator, shaping the clarity and style of the first proposal and continuing today to serve as advisor on the project

I sincerely thank the students and programmers who have made so many significant contributions to the NEOMYCIN and HERACLES programs (chronologically): Reed Letsinger, Bob London, Conrad Bock, Diane Warner Hasling, David Wilkins, Glenn Rennels, Mark Richer, Tim Thompson, Steve Barnhouse, Arif Merchant, David Leserman, and Naomi Rodolitz Victor Yu, M D. helped me understand MYCIN in the early years I am especially grateful to the late Tim Beckett, M D for showing me what a good physican teacher can do Curt Kapsner, M D , John Macias, and Bevan Yueh have enthusiastically collaborated in our recent work

The project was funded jointly by ONR and the Army Research Institute during the period 1981–1985 (ONR contract N00014-79C-0302) Computational resources have been provided by the Sumex-Aim National Resource (NIH grant RR00785). Organization and maintenance of computer resources in the Knowledge Systems Laboratory is managed by Tom Rindfleisch Continuing research is supported in part by ONR contract N00014-85-K-0305 and by a grant from the Josiah Macy, Jr., Foundation

This paper is based on a talk I gave at the Computer Science Colloquium at Carnegie-Mellon University on 19 November 1985 I thank David Evans for arranging to have the talk taped and Rene Bornstein for her accurate transcription. Many useful revisions were suggested by Steve Barnhouse, Ted Crovello, Bob Engelmore, and Mark Richer Special thanks to Susan King for TEXing the final copy.

References

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

Anderson, J. R , C F Boyle, & G. Yost. 1985. "The Geometry Tutor." In *Proceedings of the International Joint Conference on Artificial Intelligence 9*, vol. 1. Los Altos, California: William Kaufmann, Inc., 1-7.

Brown, J. S 1983 "Process Versus Product—A Perspective on Tools for Communal and Informal Electronic Learning." In *Education in the Electronic Age.* New York: Educational Broadcasting Corporation, WNET

Brown, J. S., A Collins, & G. Harris, 1977. "Artificial Intelligence and Learning Strategies " In O'Neill (Ed), *Learning Strategies.* New York: Academic Press.

Carbonell, J. R. 1970. *Mixed-Initiative Man-Computer Instructional Dialogues* Tech. Rep 1971, Bolt Beranek and Newman.

Clancey, W. J. 1979 "Transfer of Rule-Based Expertise Through a Tutorial Dialogue." Ph D diss , Stanford University

Clancey, W J. 1979 "Dialogue Management for Rule-Based Tutorials " In *Proceedings of the International Joint Conference on Artificial Intelligence 6, Tokyo, Japan.* Los Altos: William Kaufmann, Inc , 155-161.

Clancey, W J 1979 "Tutoring Rules for Guiding a Case Method Dialogue." *The International Journal of Man-Machine Studies.* 11: 25-49 (Also in Sleeman & Brown, (Eds), *Intelligent Tutoring Systems* New York: Academic Press)

Clancey, W J 1982 "GUIDON " In A. Barr & E. A. Feigenbaum (Eds), *The Handbook of Artificial Intelligence* Los Altos, CA: William Kaufmann, Inc.

Clancey, W J 1983 "The Epistemology of a Rule-Based Expert System: A Framework for Explanation " *Artificial Intelligence* 20(3), 215-251

Clancey, W. J. 1983. "The Advantages of Abstract Control Knowledge in Expert System Design " In *Proceedings of the National Conference on Artificial Intelligence, Washington, D C* Menlo Park, California: American Association for Artificial Intelligence, 74-78

Clancey, W. J. 1983. "Communication, Simulation, and Intelligent Agents: Implications of Personal Intelligent Machines for Medical Education." In *Proceedings of AAMSI-83.* 556-560.

Clancey, W J. 1984 "Teaching Classification Problem Solving (Abstract)." In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society, Boulder, Colorado.* New York: Cognitive Science Society, 44-46.

Clancey, W J 1984. "Methodology for Building an Intelligent Tutoring System " In Kintsch, Miller, & Polson (Eds.), *Method and Tactics in Cognitive Science* Hillsdale, New Jersey: Lawrence Erlbaum Associates, 51-83

Clancey, W. J 1984 "Acquiring, Representing, and Evaluating a Competence Model of Diagnosis " HPP Memo 84-2, Stanford University, February (To appear in M. Chi, R. Glaser, and M Farr (Eds), *Contributions to the Nature of Expertise*)

Clancey, W. J. 1984 "Knowledge Acquisition for Classification Expert Systems " In *Proceedings of ACM Annual Conference* New York: Association for Computing Machinery, 11-14.

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

Clancey, W J. Forthcoming "Representing Control Knowledge as Abstract Tasks and Metarules." In M. J. Coombs & L. Bolc (Eds), *Computer Expert Systems* New York: Springer-Verlag

Clancey, W J. 1986 *The Science and Engineering of Qualitative Models.* KSL Report 86-27, Stanford University

Clancey, W. J & C Bock 1982 *MRS/NEOMYCIN: Representing Metacontrol In Predicate Calculus.* HPP Memo 82-31, Stanford University.

Clancey, W J. & B. G. Buchanan. 1982 *Exploration of Problem-Solving and Tutoring Strategies: 1979-1982* Tech Rep STAN-CS-82-910, HPP 82-8, Stanford University

Continued on Page 187.