4

March 1986

Report No. STAN-CS-87-1169
Also numbered KSL-85-32

Applying a Qualitative Modeling Shell to Process Diagnosis: The Caster System

by

Timothy F. Thompson and William J. Clancey

Department of Computer Science

Stanford University
Stanford, CA 94305

87 [illegible]

Applying a Qualitative Modeling Shell to Process Diagnosis: The Caster System

by

Timothy F. Thompson, Westinghouse R&D Center

William J. Clancey, Stanford University

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

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

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

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

The purpose of knowledge engineering is to develop partial qualitative models for solving practical problems. These models--called knowledge bases in expert systems--must have appropriate diagnostic knowledge to deal with the real-world problems.

In general, solutions to diagnostic problems can be either selected from a set of pre-enumerated alternatives (for known conditions) or constructed (for novel problems or those that combine multiple, interacting disorders in an unforeseen way). While engineering design is often thought of as a constructive problem-solving process, diagnosis is typically thought of as a selection or classification problem. But the solution method is not inherent in the task itself. Instead, it depends on the problem solver's previous knowledge, requirements for customization, and the like.

For example, engineering design may involve selection from among competing possible designs (especially in the case of the experienced designer), and diagnosis sometimes requires consideration of the system's structure and function. ~~[6]~~

Nevertheless, useful programs can be developed that solve diagnostic problems by selection alone. We believe that starting with a well-defined classification procedure and a relational language for stating the classification model eases the development of a program that diagnoses by selection. To test this thesis, we built an expert system, called Caster, that addresses a particular diagnostic problem: malfunctions in industrial sandcasting. Our goal was to demonstrate that these control structures, developed for a medical diagnosis problem, are general and applicable to engineering applications.

Caster uses the Heracles qualitative modeling environment, a shell generalized from the Neomycin medical diagnosis system. Heracles provides a well-defined diagnostic procedure that structures inference according to a pattern called heuristic classification, and it provides a relational language to state knowledge of taxonomies of malfunctions and abnormal state transitions. [5]

History

The Mycin system, [11] developed in the mid-1970s, showed that it was possible for a computer to diagnose and treat infectious diseases with the accuracy of a human expert. [2] Subsequent research has sought to improve understanding of the nature of domain knowledge and the expert's inference process. One line of research resulted in Neomycin, a reconfigured and extended version of the original Mycin system. [4]

Neomycin's principled representation of domain knowledge is in terms of disease taxonomies, causal networks of abnormal states, and hierarchies of findings such as measurements, general observations, and disease symptoms. Its design also explicitly represents control strategy so that the system can articulate its diagnostic strategy to users, and so that, in a tutorial setting, the system has some basis for recognizing the logic behind student behavior. [4]

Consequently, Neomycin's knowledge base consists of both domain-level rules about medical disorders and control rules that express techniques to select and apply domain rules as human experts would. For efficiency, these control rules can be precompiled into procedural code and can be interpreted by an explanation facility to teach the system's diagnostic strategy.

Neomycin's control knowledge (a rule set that heuristically classifies findings, abnormal states, and diseases) evolved concurrently with its domain-specific knowledge base. The set of control rules developed for Neomycin has evolved into a cohesive whole characterized as heuristic classification. Knowledge engineers have used this control strategy in several expert systems, including Mycin, Puff, Sacon, the Drilling Advisor, Grundy, and Sophie III. The strategy was *not* explicitly represented, so the systems could not reason about their actions.

This is the crucial difference between Heracles and, for example, Emycin, which does not have a heuristic classification control strategy built in. Thus, Emycin users implicitly build a heuristic classification control strategy into their domain rules, and every system designer (knowledge engineer) must repeat this task. Clearly, this is a significant obstacle to efficient knowledge acquisition -- one Heracles can overcome.

Heuristic classification

Simple classification problem-solving involves selecting from a set of pre-enumerated solutions. These solutions are often organized hierarchically, and classification consists of matching observations about an unknown entity against features of known solutions.

In heuristic classification, solutions and solution features may be matched *heuristically*, by direct, nonhierarchical association with some concept in *another* classification hierarchy. For example, Mycin does more than identify an unknown organism in terms of visible features of an organism: Mycin heuristically relates an abstract characterization of the patient to a classification of diseases.

Figure 1 shows the inference structure schematically. It shows that heuristic classification consists, essentially, of data abstraction, heuristic match, and hypothesis refinement. Basic observations about the target system are abstracted into feature categories according to definitional relationships, qualitative relationships between numeric measurements and nonnumeric conceptual descriptions, and generalizations between classes of findings and their subtypes.

Inferential leap. From this hierarchy of data abstractions, the heuristic classification problem solver makes a great inferential leap via heuristic or causal relations to a hierarchy of solution classes. A heuristic relation is uncertain, often derived empirically from the problem solver's (such as a physician's) experience.

A heuristic relation typically takes the place of a network of causal relations between problem features and solutions because the causal relations are unobservable, poorly understood, or invariant for most cases. After a heuristic match or causal propagation suggests a general malfunction hypothesis, the hypothesis must be refined according to the distinctions relevant to fixing the target problem (that is, prescribing therapy).

Refinement goal. The goal of refinement is to confirm or rule out competing specializations of the general hypothesis at each level in the malfunction taxonomy by requesting specific findings or tests. The result of this refinement is the specific cause of the general malfunction.

For example, if a heuristic match suggests a general hypothesis (solution class), a heuristic classification system begins to explore and refine the classification by replacing the hypothesis on the system's current list of potential malfunctions (called the differential) with known subtypes. The system tries to gather more evidence to confirm or rule out each competing hypothesis. The system repeats the refinement phase for each subtype.

The classification hierarchies are not always trees, as generally portrayed for simplicity in this article. They may be "tangled" structures with some concepts having multiple parents. While it is convenient to think of heuristic classification as ordered steps of data abstraction, heuristic match, and hypothesis refinement, the heuristic classification model actually makes no claims about the execution order.

Depending on the needs of the application, the diagnostic procedure can be organized to reason, for example, either forward from observations to solutions or backward by hypothesizing solutions and then asking for findings that either confirm or refute the hypotheses.

The heuristic classification strategy implemented in Heracles is opportunistic: The system pursues lines of reasoning as they are suggested by new findings. The system simply does not do all the data abstraction, then all the heuristic matching, and then all the hypothesis refinement. Instead, the refinement process may suggest the need for new data (to confirm or refute a hypothesis), which in turn might trigger further data abstraction, and so on.

Heracles shell

The Heracles system is a comprehensive environment for building heuristic-classification qualitative models of engineering problems. This environment includes an explanation facility, [7] an interactive, graphics-based display developed for debugging and teaching, [10] and a graphics-based knowledge editor designed for the Caster system.

These facilities give the knowledge engineer a well-defined relational language to express knowledge about classes of solutions and their features and specializations, about classes of findings, and about networks of causal relationships among abnormal states.

We generalized Heracles from Neomycin the same way Emycin was generalized from Mycin. [12] We removed all the domain rules and domain parameters from Neomycin and generalized any references to medicine in the control rules. Figure 2 shows Heracles as the core of the Neomycin and Caster systems. The relational language for qualitative models is the same. The classification procedure indexes the model in terms of these relations.

Implementing classification. Part of the key to efficient model design for selection problems is having a heuristic-classification control strategy already built into the problem solver. If the knowledge engineer can solve a problem with this computational method -- and experience shows it to be extremely general -- his job will be much easier because the vocabulary for stating the qualitative model and the procedure for interpreting it during a consultation are supplied in advance. The knowledge engineer is thus freed to concentrate on defining the basic terms and their relationships in the target domain.

In Heracles, control rules are grouped into tasks like explore-and-refine a general hypothesis, group-and-differentiate the current hypotheses in the differential, process a (new) finding, and so on. There are 29 tasks and 75 control rules. (These control rules are also referred to as metarules since they are essentially rules which reason about the application of domain-specific rules or, more generally, what information to gather, and what assertions to make.)

Each task is associated with a short, ordered set of control rules. These rules are often applied iteratively according to the specification of the given task. Thus, heuristic classification in Heracles is highly structured and has relatively few steps or methods to achieve any one task.

Figure 3 shows an example of one of the control rules Heracles uses to process input data. Roughly translated, it says, "If the current task is forward-reason, and if a newly entered piece of data has not yet been clarified, then invoke a subtask called clarify-finding to further classify the data."

For example, if Neomycin learns that the patient has a headache, one of the possible choices it has is to ask more specific questions (such as how long, how often, or exactly where these headaches occur) that further clarify the initial, general finding of headache.

It is exactly this collection of tasks and metarules developed in Neomycin that we use in Caster, although not all of them apply. For instance, those rules pertaining to the distinction between circumstantial evidence and laboratory data are superfluous in Caster because all the data in Caster is circumstantial.

Knowledge-expression language. Another part of the key to efficient model design for selection problems is a well-defined relational language in which the knowledge engineer can express knowledge of the domain. In Heracles, the fundamental terms (domain parameters) include findings, descriptions of abnormal internal occurrences (for simplicity we will call them states), and treatable causes. The fundamental relations include heuristic rules expressing a cause and effect (or causal) relationship, subsumption relations, taxonomic relations, and qualitative abstraction rules.

The possible relations between domain parameters that we need to describe the Caster system are as follows. Findings can subsume other findings (for example, since brain surgery is "a kind of" surgery, surgery is said to subsume brain surgery). Likewise, abnormal states can subsume other abnormal states. Abnormal processes are grouped into a classification hierarchy of diagnostic hypotheses by links that express a type/subtype or taxonomic relationship.

Findings and diagnostic hypotheses may be related directly by heuristic rules or by causal networks of abnormal states. Thus, findings can be caused by abnormal states, abnormal states can be caused by other abnormal states, and abnormal states can be caused by a particular hypothesized malfunction in a class of malfunctions. In general, diagnostic hypotheses are types of malfunctions endemic to a specific domain.

In our specific application of heuristic classification to diagnosis of processes, the disorder classification models the process in terms of what can go wrong at particular stages. Each terminal node specifies a model of the world in terms of symptoms and causes, and each terminal node exists because it is a treatable cause (that is, it can be heuristically related to fixes).

Moreover, each fix is a change to physical system functions and processes. Thus, our goal is to construct a qualitative model that will fix a physical system, not a model that diagnoses a system for its own sake.

Tools to express knowledge. The ability to efficiently build a useful knowledge base partly depends on the well-known expert system tenets: The interface should be easy to learn, and easy to use. The Heracles user interface has a graphics-based display facility and a graphics-based knowledge editor. The editor is mouse- and window-driven, and its main menu is displayed as a pop-up window that lets the knowledge engineer move a node, add a node, delete a node, add a link, or delete a link (see Figure 4).

Nodes correspond to domain parameters such as findings, abnormal states, and malfunctions. Links refer to relations between parameters, such as causal rules, subtype relations, and subsumption relations.

An important feature of the knowledge editor is its knowledge of which domain parameters can be associated by which relations. For example, there are a number of slots associated with causal rules, and each slot may take any of a number of values. The knowledge engineer must only specify nodes and links between them -- the knowledge editor will determine which slots and values are appropriate, and then fill them in automatically.

Thus, the knowledge engineer doesn't have to know how to fill all the appropriate slots in all the appropriate schemata each time he specifies a domain relation. He doesn't even have to know that these slots exist. The knowledge editor makes the job of entering new knowledge much easier. It also makes it easier for the knowledge engineer to see, literally, how the changes he makes will affect the old knowledge.

Sandcasting domain

Sandcasting is a metalworking process used to make all sorts of objects from crescent wrenches to V8 engine blocks to ocean liner propellers. It involves packing a two-part box, called a flask, full of sand around a pattern of the object to be cast. In Figure 5, the casting is of an L-shaped object. The pattern is removed by separating the box along a parting line. The dimensions of the flask can be from about a foot square to 20 or 30 feet square, depending on the size of the object to be cast.

Molten metal is poured down a large tapered hole, called a sprue, into a cavity, called a sprue well, that cuts down on the turbulence caused by pouring and that allows impurities to settle before flowing into the mold cavity. The metal flows into the mold through runners that open onto the mold through gates.

The placement and size of the pattern equipment -- sprues, runners, and gates -- is critically important. For example, if the gates are too small, the metal may not fill the mold before beginning to solidify, thus cutting off flow into the mold prematurely. On the other hand, if the gates are too large, impurities can wash into the mold, causing structural and surface defects, like cracks and bubbles, in the casting. There are also many other considerations in pattern equipment design.

As the metal cools, its volume is reduced because molten metal has more volume than solid metal. As the metal solidifies, it must be kept under pressure and more molten metal must be available to replace the lost volume. Other funnel-shaped holes called risers, which fill with metal during pouring, take care of this. The force of gravity on the metal in the risers can exert enough pressure to keep the casting from developing shrinkage cracks.

This qualitative physical description of sandcasting forms the foundation to understand the techniques in diagnosing sandcasting malfunctions. The problem is how to get a computer to understand sandcasting so that it can diagnose malfunctions.

The malfunctions we intend Caster to diagnose are the common and repeatable failures the quality control engineer faces every day. For example, the temperature of the molten metal, the pattern equipment, and the molding sand can vary from casting to casting. The pattern equipment will gradually wear away with time. The molding sand may contain varying amounts of moisture. Levels of impurities in the molten metal will vary. All these variables and many more will cause repeatable sandcasting malfunctions.

Knowledge acquisition experiment

The knowledge bases for most expert systems depend very heavily on substantial input from domain experts. Unfortunately, the time of the skilled diagnostician is usually in great demand, so the time needed to build a quality knowledge base clashes with the need to have the expert fight fires elsewhere at the same time.

On the other hand, reference material for a given domain is often readily available -- but it is generally recognized that written references are a poor substitute for the domain expert.

Because the demand for the domain expert's time will always exceed the time available and because domain experts frequently consult reference works with good results, a fruitful research area would be trying to improve domain-knowledge acquisition techniques for written reference material.

Firmly entrenched procedure. One important reason an expert can assimilate new knowledge from reference manuals is that he starts with a firmly entrenched problem-solving procedure and a solid understanding of the fundamental terms and relations in his domain. We hypothesized that if we could equip a novice in the domain with a well-defined problem-solving procedure and a relational language for the knowledge representation, perhaps he, too, could efficiently construct a qualitative model of the domain from written reference materials.

Earlier research in studying a variety of knowledge bases, which showed that heuristic classification is a robust and well-defined problem-solving strategy for selection problems, seemed to endorse this hypothesis. Furthermore, there is considerable psychological support for heuristic classification as a model of how experts use experiential knowledge of familiar problem situations and solutions. [5]

To test our hypothesis, we chose an experiment in knowledge acquisition where we attempted to become, in a limited sense, our own domain experts by studying many textbooks and manuals on diagnosing sandcasting malfunctions, rather than extensively interviewing experts.

We tried to use heuristic classification and Heracles's relational model language as the foundation for our own emerging mental qualitative model, and as a framework to guide the instantiation of Caster's computer model. We relied heavily on the casting handbook *Analysis of Casting Defects* [1] and referred to experts only on difficult or unclear matters.

Developing a qualitative model. There are about 15 major classes of sandcasting malfunctions. We initially concentrated on the kinds of sandcasting malfunctions that cause shrinkage cavities. A shrinkage cavity is the result of letting the casting cool without forcing new metal into the mold to offset the reduction in volume as the metal solidifies.

A shrink is an observable defect that would be considered a general finding in Caster as much as a headache is a finding in Neomycin. It is then possible to clarify that finding according to how long it has been occurring, what percentage of castings exhibit this defect, and the like.

Original plan. In our original plan, after establishing the data abstraction hierarchy for shrinks, the next goal was to identify from the manual the abnormal processes that could cause a shrink and then to organize them into a classification hierarchy with general malfunctions at the top and more specific malfunctions further down.

Finally, we wanted to uncover the heuristic relations between the data abstraction hierarchy and the taxonomy of malfunctions or perhaps identify a causal network of abnormal states relating abstractions and malfunctions.

In retrospect, we might have expected the job would not be so straightforward. Although we came to the task well-armed with our explicitly represented diagnostic procedure and relational model language, the authors of the casting manual didn't have the same commitment to explicit diagnostic models.

Each sentence's mention of causality of some sort obscured problem feature hierarchies and malfunction taxonomies. The manual consists largely of independent heuristic causal relationships between observations and diagnoses. For example, the manual has many sentences like "Excessive gas producing inorganic materials can cause gas defects in castings . . . ," "Uncured shell sand will . . . cause pinholes," and "Excess moisture in green molding sand is probably the greatest cause of gas defects in cast metals."

Thus, this casting manual is loosely organized as a very long list of causal relations of the form <some malfunction> causes <some symptom>. This long list is subdivided into about 15 major sections corresponding to the major observable symptoms. Each section is divided into several subsections corresponding to the basic processes in which malfunctions arise.

General diagnostic strategy. The authors of the manual made no attempt to teach a general diagnostic strategy, probably because the assumed reader is one already skilled in the art of sandcasting diagnosis. By assuming that the reader is a sandcasting expert with his own well-developed diagnostic methods, the authors were free to provide useful reference information in a highly concentrated form that the reader can assimilate into his existing "knowledge base" as he sees fit.

Our first pass at representing this type of description resulted in a group of fragmented causal associations (see Figure 6). About all we were sure of was that shrinkage cavities were caused by a variety of problems, including inadequate supply, not enough gates and risers, risers that are too small, and gates that are too large. From this sort of description, it was not at all obvious what were treatable causes, what were internal descriptions, and so forth.

Fundamental cause. However, after much reflection and several abortive attempts at restructuring our initial list of causal associations, we realized that a fundamental cause of shrinkage cavities was inadequate supply and that the other problems mentioned, like too few gates or risers, were causes of inadequate supply.

There are essentially three reasons why the casting might be inadequately fed. One is that there simply is no metal available to feed. This can result from too few risers, risers that are too small, and similar reasons. Another cause is that the mold broke, possibly due to mishandling. Thus, the metal was available but leaked out. The third cause is that the feed is shut off, possibly because the neck of the riser is too small (that is, the metal in the small riser neck solidified, preventing an adequate flow) or the metal was not heated sufficiently.

It should be clear that we were doing more than merely translating the text in the manual into Heracles's relational model language. In studying the manual, our own qualitative model of the shrinkage cavity problem gradually emerged. Only after we really understood the causality behind shrinks could we put that knowledge in the language used by Heracles.

At the same time, however, we used the causal structure defined by Heracles to enable us to ferret out the causal networks of abnormal state transitions implicitly expressed in the casting manual.

Another important insight was the realization that the general malfunction classes in

Neomycin are fundamentally different than those found in the sandcasting domain. In medicine, we see clearly identifiable abnormal processes (such as infectious processes, vascular disorders, and traumatic processes) and abnormal agents that cause those processes. In casting, there are no analogous abnormal processes or agents. Instead, there is a clear sequence of normal processes (such as melting, pouring, and solidifying) that are affected by abnormal events (such as a mold breaking or the feed being shut off).

Partial model. Figure 7 shows a simplified version of our reorganized knowledge of shrinkage cavities which shows the normal processes of melting, molding, solidifying, and so forth, with the abnormal conditions that might occur in each process. For example, a poor jacket fit is just one abnormal condition which might arise during the normal molding process.

Because we had difficulty with our first attempt at building a qualitative model for shrinkage cavities, and because we believed we had resolved the source of that problem, we continued developing knowledge of another common type of casting malfunction: gas defects (see in Figure 8).

Significantly, it took several man-months to come to grips with the shrinkage cavities model while it took only a couple of weeks to develop a similarly complex model for gas defects.

Example session. The following is an excerpt of a script (annotated in italics) showing heuristic classification as applied to the partial qualitative model shown in Figure 7.

```
16-May-85 22:43:56
 ------CASTING-4------
Please enter information about the casting
1) Casting Type:
IRON
2) Please describe the chief complaints:
** SHRINKAGE-CAVITIES
**
```

Caster begins by applying the heuristic rule suggesting inadequate supply.

```
DIFFERENTIAL: (INADEQUATE-SUPPLY 800)
```

Caster is exploring and refining the differential. Inadequate-supply has been replaced by metal-leak, feed-is-shut-off, and no-metal-to-feed. Metal-leak suggests a broken-mold, so Caster inquires whether there is evidence of a runout.

```
3) Does Casting-4 have a runout?
** N
```

Caster has looked at feed-is-shut-off, which suggests that fillets may be too small, so the system asks if the shrink is at a corner, which would be strong evidence for fillets-too-small.

```
4) Does Casting-4 have a shrink at a corner?
** Y
```

Finally, Caster looks for evidence that there is no metal available to feed, namely that there is a high reject ratio.

```
5) Does Casting-4 have a high reject ratio?
** N
DIFFERENTIAL: (METAL-LEAK 100) (FEED-IS-SHUT-OFF 100)
(FILLETS-TOO-SMALL 700)
```

This example shows how heuristic classification can reduce the amount of search required by ruling out some lines of reasoning while at the same time promoting a more focused and therefore more natural interaction with the user. The example also illustrates hypothesis refinement and the opportunistic style in which heuristic classification is implemented.

Inadequate-supply is replaced in the differential by its three children, metal-leak, feed-is-shut-off, and no-metal-to-feed. Each of these more specific hypotheses suggests hypotheses elsewhere in the malfunction taxonomy. Caster adds those hypotheses to the differential and tries to explore them along with the subtypes already in the differential by asking for confirming evidence which could in turn trigger further data abstraction. (Their children do not come into play in this example, and are therefore not shown in Figure 7.)

The next step in our development of Caster would be to extend and validate Caster's domain knowledge by using the system in the context of real-world problems. We are ready to perform such an analysis, but because the foundry has only very recently begun recording malfunction case histories, we cannot yet get the required data.

Experts examining traces of Caster working on hypothetical cases confirm that Caster does appear to make the correct diagnoses. This is important to us since one of our goals was to determine how well heuristic classification can be applied to nonmedical domains requiring causal reasoning.

Observations on the experiment. Although virtually all the well-known knowledge acquisition systems are designed specifically for human knowledge sources, we were able to show that Heracles, using an explicitly implemented heuristic classification control strategy and a relational language for building qualitative models, can build partial qualitative models for the sandcasting domain from diagnostic manuals.

Novice sandcasting diagnosticians such as ourselves could succeed in building Caster because the Heracles framework helped us understand the domain by organizing the knowledge in the reference manuals so it could be used for diagnosis. Without it, we would still be staring at our original, overwhelming, unconnected, and not very useful list of causal relations from the manual.

The casting manual told us a lot of things about the casting domain, but it didn't tell us what to do with this knowledge. It didn't explain that some observations subsume other observations. It didn't explain that general malfunction classes can be refined into more specific hypotheses by asking specific questions. Heracles provided this vocabulary.

After we realized that sandcasting diagnoses are best characterized as a sequence of normal processes affected by abnormal events, we were able to use the casting manual efficiently to build partial qualitative models to diagnose sandcasting problems.

Understanding the different ways diagnostic hierarchies of processes can be organized is an important result of this research. For example, notice that the normal processes of sandcasting are subject to a temporal ordering. First we design the pattern, then build the mold, and then melt and pour the metal. Finally the metal solidifies.

A problem may occur in one process (a mistake in pattern design) but a symptom may not appear until several processes later (a shrinkage cavity). This provides a simple diagnostic heuristic we could add to Heracles -- moving from right to left in the tree of disorders (or backwards in time from effects to causes). This heuristic very nicely shows how the representation for causal processes relates to the diagnostic control rules.

Knowledge acquisition heuristics

Expert system researchers have reached the point of focusing less on "How do we build expert systems?" and more on "How do we build expert systems right?" For example, how do we design expert systems that can be better maintained, that can explain their reasoning to users, and that are easier to build?

Caster's purpose was less to demonstrate that we could build the system than it was to understand how starting with a relational vocabulary for a qualitative model influences knowledge acquisition. One result was the identification of knowledge acquisition heuristics that are new precisely because they view knowledge acquisition in terms of constructing a qualitative model.

These heuristics apply not only to diagnostic systems but also to qualitative modeling for design, modification, monitoring, and control of physical systems in general. These knowledge acquisition heuristics extend and refine the crucial questions Buchanan et al. [8] list: "What are the important terms and their interrelations? What does a solution look like and what concepts are used in it? How are objects in the domain related?" Indeed, these are crucial questions. However, identifying these questions is not nearly as difficult as answering them.

We used Heracles's relational language to help us identify the fundamental terms and relations in sandcasting. We identified the important substances in our domain, including molding sand, water, various gases, and iron. We identified the important processes that act on these substances, including melting metal, designing a pattern, building a mold, and pouring the metal.

We constructed classification hierarchies for each of these substances to define the causes of abnormal events that disrupt these normal processes. We identified high-level descriptions of the symptoms of these problems and then constructed abstraction hierarchies for each by considering the possible types of information characterizing each general finding. (That is, we looked for generalizations, definitions, and qualitative abstractions yielding the general findings.)

The last step was determining the heuristic relationships between elements of the data abstraction hierarchy and the malfunction taxonomy. Relating general classes, instead of specific findings and hypotheses, aids learning because many specific relations can be understood as special cases of a general rule.

Thus, the hierarchical structure of domain knowledge -- the categorization of processes, substances, and the like promoted by the heuristic classification model -- is especially beneficial to knowledge acquisition.

Step-by-step procedures. Our experience with Caster further shows that this step-by-step procedure for qualitative model building is generically applicable to selection problems:

- Describe abnormal events in terms of the overall stages of the manufacturing process.

Ease the categorization of processes and abnormal events by classifying abnormal events in terms of the processes they occur in. For example, the problem of inadequate supply occurs during the solidification process. A broken mold is the result of a flaw in the molding process.

- Identify abnormal properties of substances -- then look for causes.

Before thinking about causality, think of all the possible manifestations of a type of problem, and then establish causes. For example, metal contamination is a serious problem in casting. There are a number of possible contaminants, including aluminum, silicon, and phosphorus. After identifying as many contamination problems as possible, consider how each type of contamination might occur and how the problem would manifest itself. This method is much easier and more complete than trying to identify the many causes of metal contamination without first considering what those contaminants might be.

- Identify possible malfunctions, determine the corrections for those problems, and then translate those relationships to causality.

Perhaps using case histories, figure out what action is needed to fix each malfunction and translate that knowledge into a causal representation. For instance, for the feed shut-off problem, we ask what possible changes could remedy the situation. The answers include making the gates bigger, the neck of the riser bigger, the fillets larger, and the metal hotter. All these corrections appear in the causal model for shrinkage cavities shown in Figure 7.

- Establish causality between findings and malfunctions, then expand intermediate levels.

After establishing a causal chain between findings and malfunctions, go back and look for additional hidden causality between the links in the chain. We found that it is quite natural for the expert to leave out levels of causality because he knows the domain. This was frequently the case in the casting manual. By making implicit relations explicit (as measured by need in actual cases), knowledge can be organized to use search-saving strategies such as explore-and-refine.

- Express causality in, for example, natural language first, then translate it to an expert system's format.

Start by ignoring the expert system entirely and concentrate on expressing causality in a convenient way. Insisting that the knowledge engineer express his rules in a particular expert system's format risks paralyzing the engineer with unnecessary and orthogonal analysis of issues of little concern to him at the moment.

The Caster knowledge-base editor made this even easier.. It very naturally expresses parameters and the relationships between them without getting bogged down with unnecessary implementation details. The system implementation should be as transparent as possible to the knowledge engineer (and we expect this would be especially important for the domain expert).

Knowledge acquisition bottleneck

Many expert system researchers over the past 10 to 15 years have rallied behind the battle cry "Knowledge is power!" The prototypical expert system architecture emphasizes a simple inference engine that manipulates data in a knowledge base consisting of declarative rules and facts. The central idea is that the programmer is freed, by coding knowledge as declarative rules, from the constraints imposed by more traditional software languages. The programmer need not worry about each rule's context because the inference mechanism applies a rule only when its preconditions are satisfied.

This approach works in small domains but is very inefficient in real-world-scale applications. The two main problems are how does the system figure out which rules are applicable and which of those rules does it apply first?

As a result, virtually all large-scale expert systems either embed control knowledge in the domain knowledge or make the simple inference engine more complex. Often, the two approaches are mixed.

In either case, the burden of organizing system knowledge for efficient and correct use falls on the person who creates and maintains the knowledge base: the knowledge engineer. This job is extremely difficult, and the inability to encode knowledge representations except through this slow process is often referred to as the knowledge acquisition bottleneck.

There is widespread belief in artificial intelligence circles that this is the last nut to crack -- then expert systems will truly flourish in the commercial world.

However, one might argue that this intuition is overly simplistic -- what's really happening is that all the problems associated with traditional programming have been pushed onto the knowledge engineer. The Mycin system, for example, depended heavily on the skill of the knowledge engineer, who coded (in an unobvious and implicit way) virtually all the structure and strategies for using domain knowledge. [3]

As another example, most OPS5-based systems, like R1, [9] use some domain rules only to create purely contextual assertions that control the application of other rules. The basic idea is to partition the domain into a large number of small subproblems so only a limited number of rules need to be considered at any one time. Again, the burden is on the knowledge engineer to organize domain knowledge so that this partitioning is possible.

Thus, the knowledge engineer who builds knowledge bases for real-world expert system applications is still programming in very much the traditional sense of the word. In fact, his job is now even harder, both because he has to implement control strategies in languages that don't explicitly support control constructs and because people now expect his software to solve more difficult problems.

Thus, an expert system is by no means a software panacea. Someone, somewhere, has to do the difficult job of organizing domain knowledge efficiently to handle the target application correctly. Whether this organization is implemented in a standard procedural program or by a knowledge engineer in an expert system's rule base, it is a difficult and expensive job.

The central problem is that typical knowledge bases consist of both partial qualitative models and embedded inference procedures. We conclude that if control knowledge recurs in various domains, as it does in problems solved by experiential classification knowledge, distilling that knowledge and stating it explicitly once saves considerable effort.

Put another way, constructing useful models is a fundamental concern for most engineering problems. Knowledge engineering specifically concerns the construction of qualitative models. However, the dictum "Knowledge is power" doesn't discriminate between the model of the domain and the inference procedure. Thus, the relation of this new technique to traditional engineering practice is not as clear as it could be.

The inference procedure of heuristic classification supplied with Heracles reduces knowledge acquisition to constructing classification and state-transition qualitative models. Since the inference procedure is general, the power of the knowledge lies in the store of previously encountered problem situations and solutions and of heuristic connections between them.

Therefore, we believe the lessons of knowledge engineering are better stated by the dictum "A qualitative model is power!"

Acknowledgments

The research described in this article was performed while Timothy Thompson was at Stanford University supported by a Westinghouse B.G. Lamme Scholarship. William Clancey's research has been supported in part by the Office of Naval Research and the Army Research Insitute under contract N00014-79C-0302 and by the Josiah Macy, Jr., Foundation. Computer facilities have been provided by the SUMEX-AIM national resource under National Institute of Health grant RR 00785.

Timothy F. Thompson is a senior engineer at the Westinghouse R&D Center. He has developed several knowledge-based systems, ranging from diagnosis of large electromechanical systems to the configuration of power distribution systems.

Thompson received his BSEE from the University of Pittsburgh and his MSEE from Carnegie-Mellon University. He is a member of the IEEE, ACM, AAAI, and ACL.

He can be contacted at the Westinghouse R&D Center, 1310 Beulah Rd., Pittsburgh, PA 15235.

William J. Clancey is a senior research associate in computer science at Stanford University's Knowledge Systems Laboratory. He has been active in expert system research since he joined the Mycin project in 1975.

Clancey received his BA in mathematical sciences from Rice University and his PhD in computer science from Stanford University. Clancey is a cofounder of Teknowledge, Inc., and an associate editor of *IEEE Pattern Analysis and Machine Intelligence.*

He can be reached at the Stanford Knowledge Systems Laboratory, 701 Welch Road, Building C, Palo Alto, CA 94304.

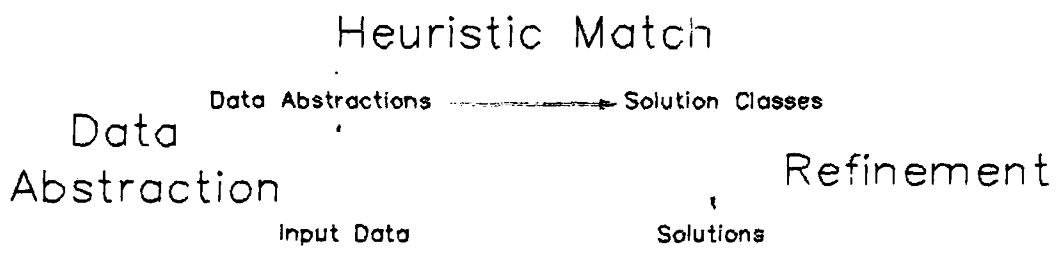

Figure 1: Heuristic Classification

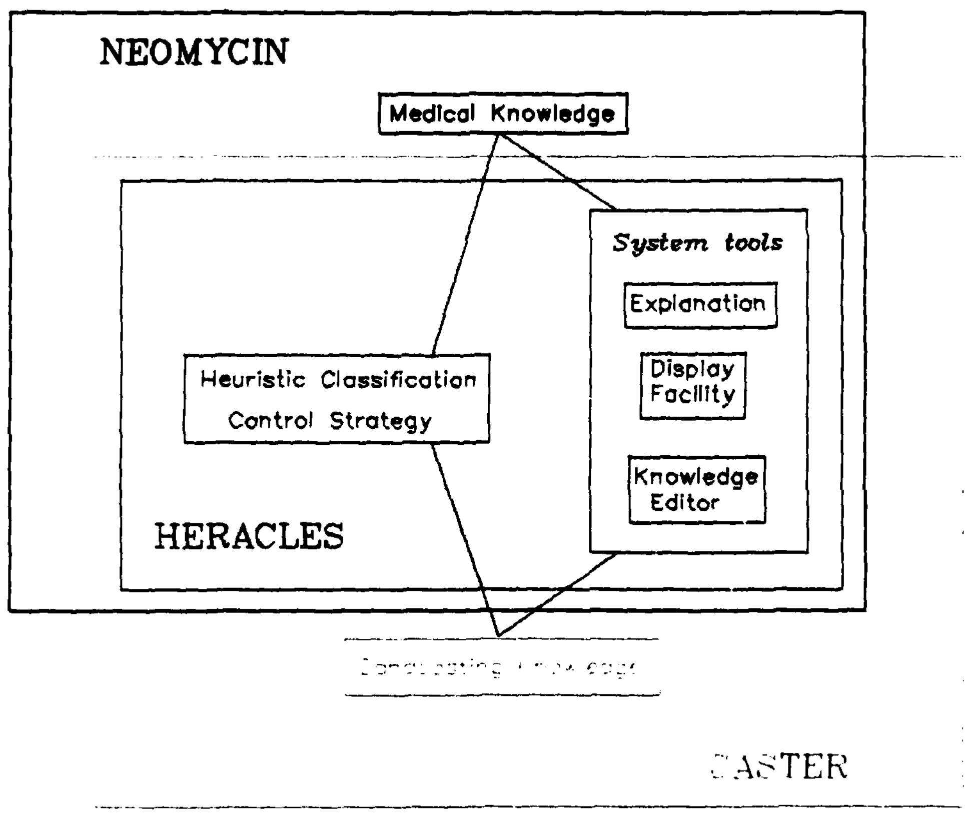

Figure 2: The Heracles Environment and the Neomycin and Caster applications

```
(META-RULE022
        PREMISE (SET-WITH-NEW (UNCLARIFIED-FINDING NEW.DATA)
                 FOCUSDATUM)
        ACTION (TASK CLARIFY-FINDING FOCUSDATUM)
        TASK FORWARD-REASON)
```

Figure 3: A sample metarule.

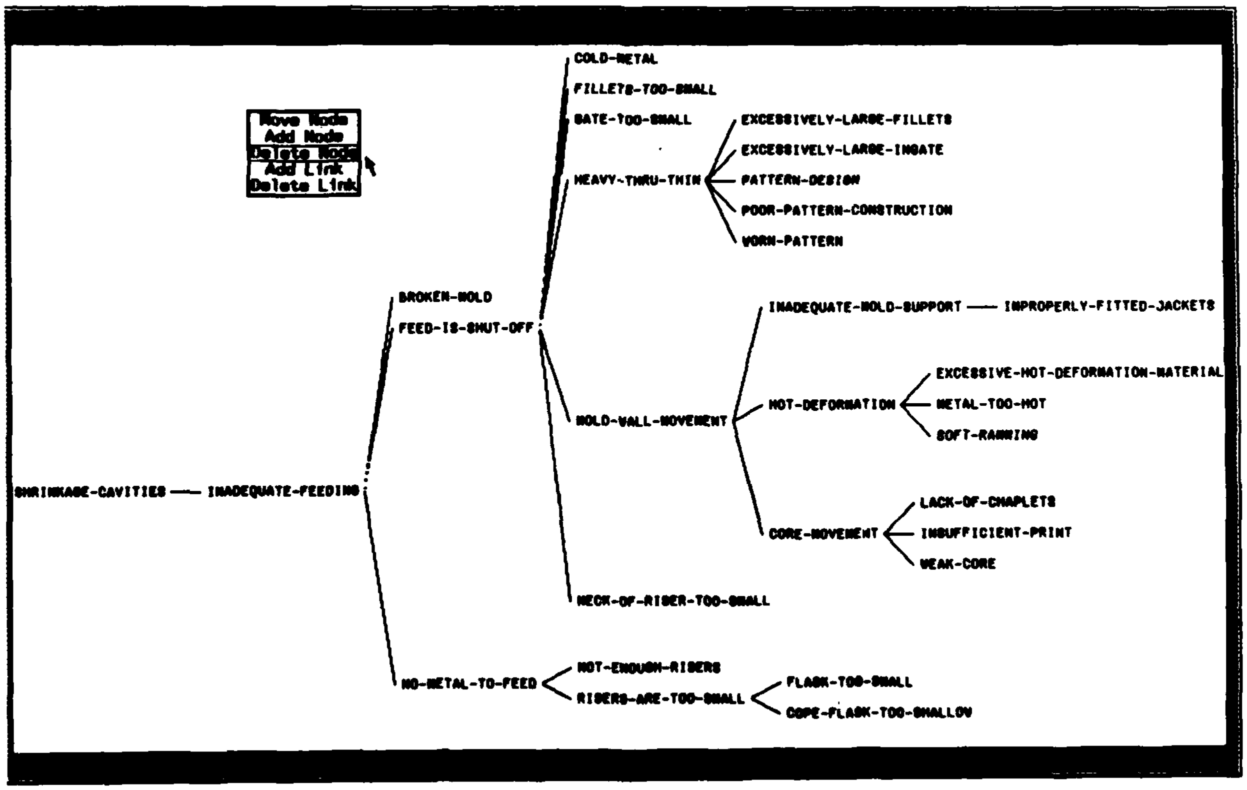

Figure 4: Using the knowledge editor.

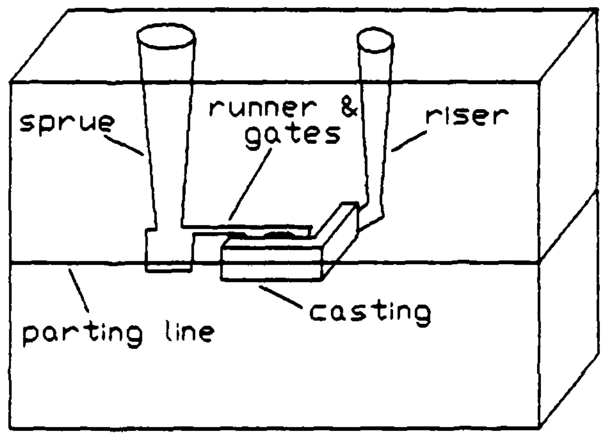

Figure 5: Illustration of sandcasting

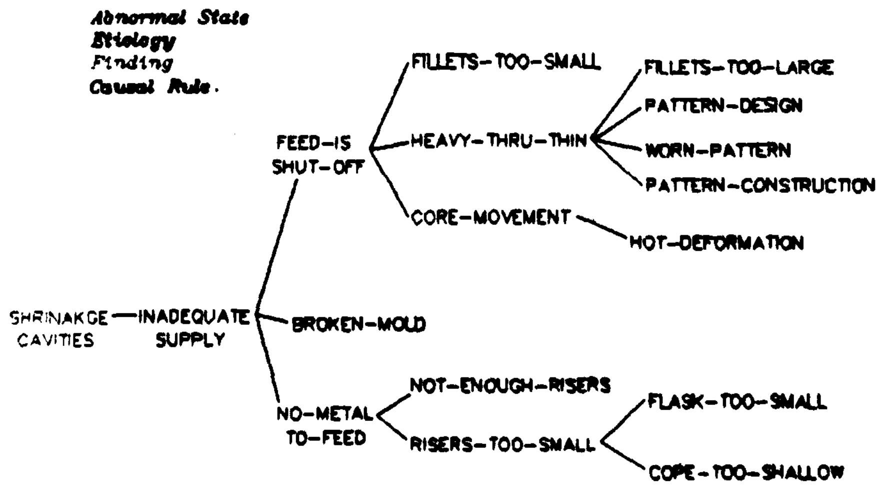

Figure 6: Causal network for shrinkage cavities.
[Shrinkage-cavities is a finding; ultimate etiologies are terminal nodes on the right; abnormal states appear as intermediate nodes linked by causal rules.]

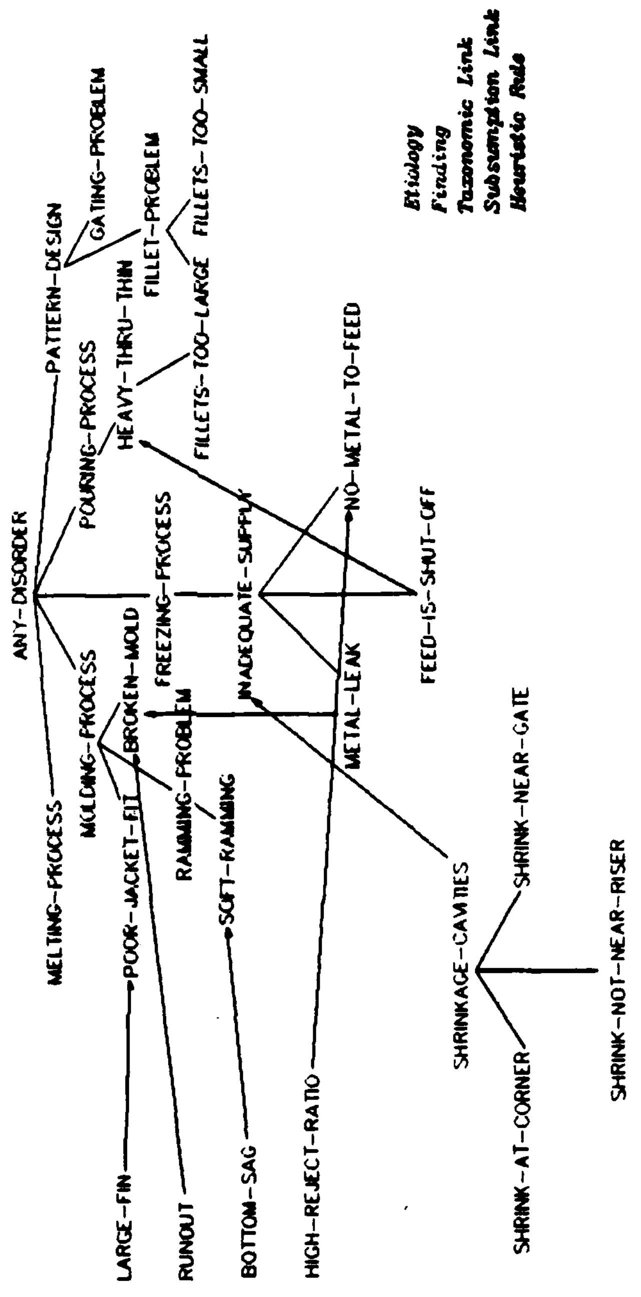

Figure 7: Knowledge of shrinkage cavities organized for heuristic classification.
[Etiologies appear below ANY-DISORDER; findings on left are linked to their causes by heuristic rules and to each other by subsumption.]

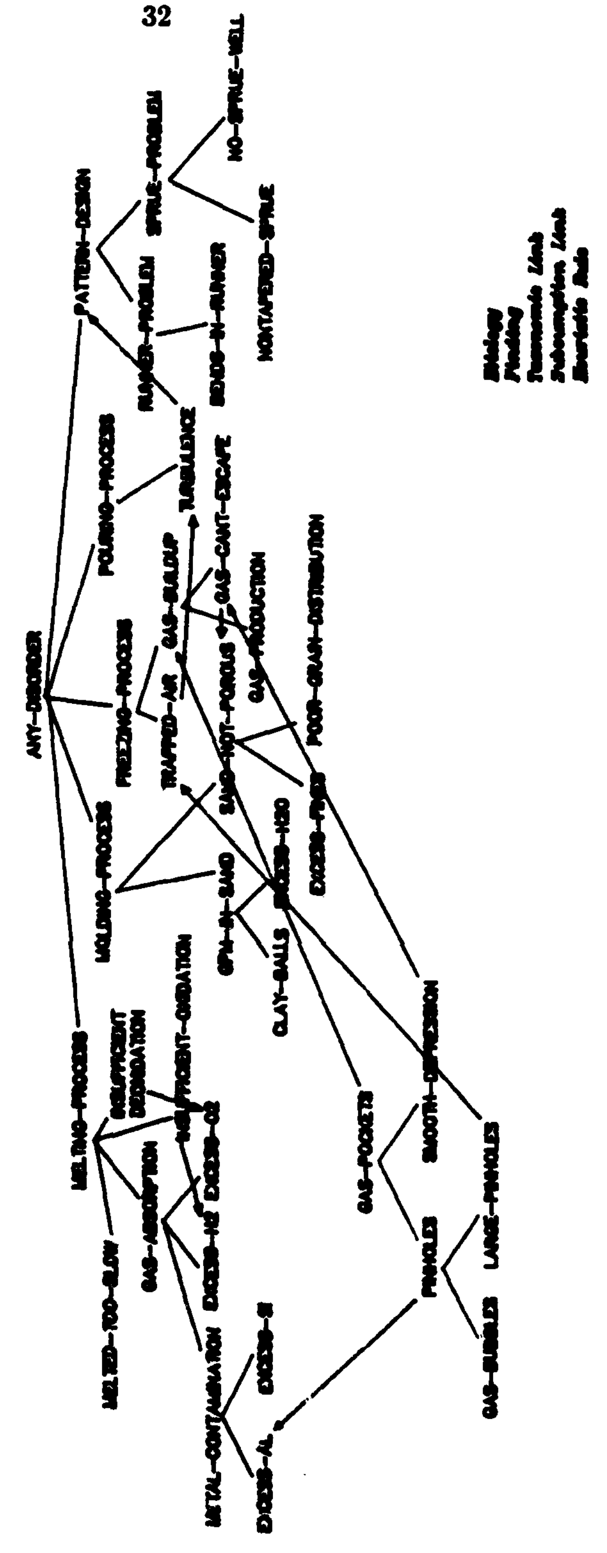

Figure 8: Partial qualitative model for gas defects.
[Etiologies appear below ANY-DISORDER; findings below are linked by subsumption; arrow lines indicate heuristic rules.]

References

[1] G.W. Anselman, et al.
Analysis of Casting Defects.
American Foundrymen's Society, Des Plaines, Ill, 1974.

[2] V.L. Yu, *et al.*
An Evaluation of MYCIN's Advice.
In Bruce G. Buchanon and Edward H. Shortliffe (editor), *Rule-Based Expert Systems*, pages 589-596. Addison-Wesley, Menlo Park, Cal., 1984.

[3] W.J. Clancey.
The Epistemology of a Rule-Based Expert System -- A Framework for Explanation.
Artificial Intelligence 20:215-251, , 1983.

[4] W.J. Clancey.
Acquiring, Representing, and Evaluating a Competence Model of Diagnostic Strategy.
Technical Report HPP-84-2, Stanford University, February, 1984.
(To appear in Chi, Glaser, and Farr (Eds.), *Contributions to the Nature of Expertise*, in preparation).

[5] W.J. Clancey.
Heuristic Classification.
Artificial Intelligence 27:289-350, 1985.
Also in *Knowledge-based Problem Solving*, ed. J.S. Kowalik, Prentice-Hall, Inc., 1985. Also KSL 85-5.

[6] R. Davis, H. Shrobe, W. Hamscher, K. Wieckert, M. Shirley, and S. Polit.
Diagnosis Based on Description of Structure and Function.
In *Proceedings of the National Conference on AI*, pages 137-142. Pittsburgh, PA, August, 1982.

[7] D.W. Hasling, W.J. Clancey, G.R. Rennels.
Strategic explanations for a diagnostic consultation system.
The International Journal of Man-Machine Studies 20(1):3-19, , 1984.

[8] F. Hayes-Roth, D. Waterman, and D. Lenat, (eds.).
Building Expert Systems.
Addison-Wesley, New York, 1983.

[9] J. McDermott.
R1: A Rule-Based Configurer of Computer Systems.
Technical Report CMU-CS-80-119, Carnegie-Mellon University, April, 1980.

[10] M.H. Richer and W.J. Clancey.
GUIDON WATCH: A Graphic Interface for Viewing a Knowledge-Based System.
IEEE Computer Graphics and Applications 5(11):51-64, 1985.
Also KSL 85-20.

[11] E.H. Shortliffe.
Computer-Based Medical Consultations: MYCIN.
American Elsevier, New York, N.Y., 1976.

[12] W. van Melle.
System Aids in Constructing Consultation Programs.
UMI Research Press, Ann Arbor, MI, 1981.

List of Figures