Cognitive Science in Medicine: Biomedical Modeling

edited by David A Evans and Vimla L Patel

The MIT Press

From The MIT Press

MITCogNet

MIT Press

0-262-05037-4

Evans, David A. and Patel, Vimla L. (Eds.)
Cognitive Science in Medicine

This book was set in PostScript^{tm} Times Roman with Donald E. Knuth's TeX and Leslie Lamport's LaTeX and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Cognitive science in medicine / edited by David A. Evans and Vimla Patel
p. cm.
Includes bibliographies and index.
"A Bradford Book."
ISBN 0-262-05037-4
1. Medicine–Decision making–Psychological aspects.
2. Physicians–Psychology. 3. Cognition. 4. Problem solving–
–Psychological aspects. I. Evans, David A. (David Andreoff), 1948– .
II. Patel, Vimla.
[DNLM: 1. Cognition. 2. Expert Systems. 3. Logic. 4. Models, Psychological. 5. Problem Solving.]
R723.5.C58 1988
610'.1'9–dc19
DNLM/DLC
for Library of Congress 87-29741
CIP

Chapter 8

GUIDON-MANAGE: Teaching the Process of Medical Diagnosis

Naomi S. Rodolitz, William J. Clancey

Expert problem solvers rely on two types of knowledge: (1) knowledge of domain facts and relationships, and (2) knowledge about how to apply these facts and relationships to solve a particular problem. Some computer tutoring systems (Wenger, 1987), as well as classroom teachers, attempt to teach both kinds of knowledge. The GUIDON-MANAGE program is designed to teach strategy through an explicit language describing the reasoning process. Specifically, GUIDON[1]-MANAGE teaches the strategic knowledge used in performing medical diagnostic consultations.

To the casual observer, a medical diagnostic consultation consists of a series of questions (Figure 8.1). Some questions will be asked in rapid succession; others will have long pauses between them. However, beneath this surface form lies a complex structure that encompasses the diagnostician's reasoning, akin to a grammar (Clancey, 1984). Clancey and Letsinger (1984) worked with physician/teacher Tim Beckett to formalize this reasoning strategy, and represented it in the expert system NEOMYCIN. GUIDON-MANAGE is part of the family of tutoring systems that work together with NEOMYCIN to help students learn both medical knowledge[2] and the process of solving diagnosis problems (Richer & Clancey, 1985; Wilkins, Clancey & Buchanan, 1985; London, 1986).

The process of structuring consultation questions is the motivating principle of the GUIDON-MANAGE system. The sequencing and relation of

[1] "GUIDON" is pronounced like "guide on."

[2] NEOMYCIN's domain is meningitis and those diseases that may be confused with meningitis.

14-May-87 10:56:00
[consultation of 1-Nov-85 10:25:36]

———PATIENT-1002———

Please enter information about the patient.

	Name	Age	Sex	Race
1)	** Susanne	44 YEARS	FEMALE	CAUCASIAN

2) Please describe the chief complaints:
** HEADACHE
** PHOTOPHOBIA
** RASHES
** FEBRILE

3) How long has Susanne had this kind of headache?
** 12 HOURS

4) How severe is Susanne's headache (on a scale of 0 to 4 with 0 for very mild and 4 for very severe)?
** UNKNOWN

5) What is Susanne's temperature (in Fahrenheit)?
** 105.8F

6) Does Susanne have a stiff neck by history or on physical?
**

Figure 8.1: Fragment from a diagnostic consultation

diagnostic questions is driven by the inherent structure of tasks, medical facts, and their dependencies. GUIDON-MANAGE is designed on the assumption that explicit teaching of diagnostic strategies will benefit learning in general (Clancey et al., 1986). Within the GUIDON-MANAGE system, the surface level of questions is stripped away, allowing the reasoning mechanism of the consultation to become visible. The student is taught a language of diagnostic tasks that can be used to describe the diagnostic process.

The goal of this project is to lay bare to the student the structure at the heart of a diagnostic consultation. This structure is comprised of a hierarchy of diagnostic tasks that takes into account relationships among medical facts and data. Psychological studies (e.g., Young's (1983) work with conceptual models and reverse Polish notation calculators) have already shown that teaching a student the underlying mechanisms of a process (often called the *black box*) is a powerful tool for instruction. Such a model establishes a framework for problem solving, general enough to adapt to new problems and guide the student's learning.

Through its explicit language of diagnostic tasks, GUIDON-MANAGE forces the student to take one step back from the surface level of asking questions. Thus the student is lead to reflect on the structure of his or her diagnostic method. In GUIDON-MANAGE, a student performs a consultation by choosing strategic tasks (e.g., *clarify a finding*) and foci (e.g., the *finding* to clarify) rather than by directly requesting data. This forces the student to consider the problem explicitly at a higher level of reasoning—one of diagnostic tasks, hypotheses, findings, and their hierarchies, rather than in terms of data collected from a random series of questions. The program responds to the student's actions by updating the diagnostic differential, supplying data (retrieved from the computer-stored patient record), summarizing conclusions made by the system, suggesting new tasks, and providing feedback.

The tutoring environment is carefully designed. There are two central considerations. First, the reactive nature of the system allows the student to learn while actively participating in a consultation. Since the system responds to each student input by updating the state of the consultation (information known, information newly derived, etc.), the student is immediately aware of the effect of a particular task. The student's perception that his or her actions have a direct effect on the state of the environment of the system has been shown to be an essential factor in keeping the student's interest (e.g., direct manipulation interfaces (Hutchins, Hollan & Norman, 1986)).

Second, the system establishes a cooperative environment between the student and the expert. The student tells the expert (the NEOMYCIN program) what actions, or tasks, to do next. The expert, in turn, carries out these tasks, offers suggestions when the student is floundering, and takes responsibility for keeping track of case-specific conclusions.

This project also examines some fundamental issues in artificial intelligence. The separation of strategic/control knowledge from domain knowledge and the modularity of the tasks and metarules in the NEOMYCIN system are tested as we allow students to choose any of a number of tasks to perform in any order, with no guarantee of the tasks that preceded them. NEOMYCIN uses a particular order in its problem solving; however, this is not the only correct strategy. GUIDON-MANAGE must therefore be prepared to accept alternate strategies, within certain bounds.

The next section briefly describes the architecture of NEOMYCIN's diagnostic knowledge and discusses some of the relevant literature. Section 8.2 describes the interface, illustrated by fragments of a GUIDON-MANAGE session. Underlying implementation details are covered in Section 8.3. GUIDON-MANAGE has been tested with a small group of medical students; Section 8.4 considers some of the observations made during these trials. Section 8.5 summarizes what has been learned and how the work might proceed.

8.1 Background

8.1.1 Relevant Research in Computer-Based Tutoring Systems

"Drill and practice" interactions were among the earliest uses of computers in education. These first computer-aided instruction (CAI) systems present a sequence of problems in a particular domain to the student and check the student's response against the answer given by the instructor. If the student's response does not match, the 'correct' answer is given, perhaps along with a canned explanation, and the system proceeds to the next problem. In some of the later systems of this genre, the instructor adds special explanations for certain incorrect responses. These systems might also give a set of extra problems to the student, depending on the misconception that the instructor believes causes a particular error. However, these systems do not explicitly represent the knowledge of how actually to solve problems, requiring the teacher's expertise to be redundantly expressed in each case or lesson.

In the past decade (Harless et al., 1971), the focus of the research in computers and education has shifted from such clever flash cards to systems that take a more active role in tutoring sessions. These systems, called Intelligent Tutoring Systems (ITS), often possess general knowledge about the domain that they teach, which allows them to generate explanations, provide definitions of terms, and usually solve the problems themselves. Some ITS systems are concerned with identifying a student's difficulties by analyzing responses (Burton, 1982; Johnson & Soloway, 1985). Others focus on taking advantage of the capabilities that the computer provides to create new environments for learning (Brown, Burton & De Kleer, 1982; Clancey, 1986).

ITS research can be roughly described along two dimensions: content and environment. The content of a tutoring system can range from electronic circuit troubleshooting to medical diagnosis to subtraction. The environment reflects the tutoring methodology within a system. The range of these environments may vary from open-ended, purely student-driven environments, such as LOGO (Papert, 1980), to systems closer to the more traditional, rigidly-structured CAI (Clancey, 1987). GUIDON-MANAGE lies closer to the LOGO end of the scale, but its cooperative nature and the inherent restrictions of the task language offer more structure to the tutoring session.

Content—Process vs. Product

The *content* of most tutoring systems has traditionally been concerned with the product of a student's actions as opposed to the process (Brown, 1985). Although many programs model reasoning strategies in order to explain a student's answer (e.g., Sleeman & Brown, 1982; VanLehn, 1983; Anderson, Farrell & Sauers, 1984), little work has been done actually to teach problem-solving processes.

The exception, which inspired GUIDON-MANAGE's design, is ALGEBRALAND (Brown, 1982), a program that *reifies* the process of problem solving in algebra. The student can reflect on what he or she did and how it affected the state of the current algebraic expression. As with GUIDON-MANAGE, ALGEBRALAND focuses on a language that the student may use to express his or her actions. Each 'task' operator can be applied to an algebraic statement so that a new expression (the end product of doing this operation) is calculated and written below the old one. Similarly, the GUIDON-MANAGE diagnostic tasks can be thought of as operating on some model of the current state of the consultation (Clancey, 1986), in which the operands are domain findings and hypotheses. For example, the TEST-*a-hypothesis* task can be applied to a particular hypothesis, such as *meningitis*.

Environments

Some of the most interesting work in ITS has been in the design of instructional environments. These environments allow students to become active participants in the learning process. Examples are programs that model the students actions; simulators; coaches; and other tools that will aid the student.

A tutoring environment can range from the Socratic dialogue of SCHOLAR (Carbonell & Collins, 1973) to the 'game' of WEST, with its built-in coach to help the student, to a simulation such as SOPHIE where the student can pose alternative hypotheses to an open-ended exploration environment such as LOGO, consisting of a simple yet powerful programming language that encourages discovery learning. Environments such as LOGO are diametrically opposed to systems that closely monitor or restrict the student's actions.

ITS student modeling components have encountered many obstacles in the quest to comprehend student actions. In some cases, the students do not have the knowledge presupposed by the system; or they don't know how to use the information they know; or their own reasoning strategy is inherently different from the system's strategy; or perhaps their objectives simply are not to learn but to play or test the tutor. Observations such as these played an important role in the decision to build GUIDON-MANAGE, at least initially, as a learning environment rather than a stricter tutoring environment. A session in GUIDON-MANAGE is still considerably more structured than a purely student-driven environment such as LOGO; however, the exploratory element is an integral part of both. Although we plan to include more traditional tutoring components in the GUIDON-MANAGE environment, the principle of allowing the student to discover on his or her own will remain intact.

Cognitive Apprenticeship and Learning Environments

In their paper on *cognitive apprenticeship*, Collins, Brown, and Newman (1986) propose six types of teaching methodology that exist within the ideal learning environment: modeling, coaching, scaffolding and fading, articulation, reflection, and exploration. In apprenticeship, a student (apprentice) is learning a skill through observation (of the mentor), through coaching (by the mentor), and through increasingly independent problem solving (by the apprentice).

Medical training through clerkships and residencies is very similar to an apprenticeship. The medical student begins his or her apprenticeship by observing a clinician performing diagnosis. The student is soon expected to assist with some small tasks during the consultation under the *coaching* of the clinician. As time progresses, the student's responsibilities increase and the clinician's support diminishes. This process of slowly introducing the student into the domain as a partner with the expert and then increasing the responsibilities and independence of the student is what Collins et al. (1986) refers to as *scaffolding and fading*. The medical students are often asked to submit a report on each consultation in which they *articulate* what they did during their session. These reports are presented to a group of physicians and other medical students who will expect the student to *reflect* on what was done in relation to comments on how one of the physicians might have handled the case.

The main difference between this type of apprenticeship and those that Collins et al. (1986) describes as a *cognitive apprenticeship* is the emphasis that the latter must place on externalizing processes that are often left implicit. Although reasoning processes are discussed to a certain extent in medical apprenticeships, the structure of these processes is not explicitly presented to the students. In an ideal cognitive apprenticeship situation, one of the first steps is for the experts to make their own models explicit. For example, the task stack in Figure 8.4 models how the expert arrived at the question "Does Suzanne have a stiff neck by history or exam?"

Exploration is another important step in the student's cognitive apprenticeship. Here the student is given a problem to solve as a catalyst for independent and creative work. Going off on a tangent at this point is not to be discouraged. These ideas can been seen in the LOGO work, where most of the learning is initiated by the student's own discovery (Papert, 1980).

Of the criteria described by Collins et al. (1986) for learning environments, exposing an expert model, scaffolding, and exploration have played the greatest part in the design of GUIDON-MANAGE. These are evident in the basic principles of the system: (1) introduce the student to a concrete language for diagnostic problem solving, (2) provide an environment in which the student can learn the meaning and use of this language through cooperative problem solving with an expert (NEOMYCIN), and (3) allow the student to explore freely the use of this language and its consequences within the environment.

Articulation and reflection were integrated into the student trials, but outside of the GUIDON-MANAGE environment itself. For example, when running a student in GUIDON-MANAGE, we often ask the student to redefine the tasks in his or her own words, or compare and contrast tasks, to help the student verbalize what is important about these tasks. Some of the later student trials involved an exercise in reflection, where the student watched NEOMYCIN solve a diagnostic problem, before or after a session with GUIDON-MANAGE, and used this as a comparison against his or her own actions.

8.1.2 NEOMYCIN and Medical Diagnosis

As was mentioned earlier, tutoring systems gain considerable leverage by including enough knowledge of the domain to be able actually to solve problems posed to the systems. NEOMYCIN serves this role in GUIDON-MANAGE. One of the most important differences between NEOMYCIN and its predecessor, MYCIN, is that NEOMYCIN has explicit knowledge about how to perform diagnosis that is separate from the medical knowledge about evidence for disease and their subtypes (Clancey & Bock, 1988).

This *strategic knowledge* is represented as a hierarchy of *tasks* and *metarules*. Each task represents an activity that is part of a medical expert's diagnostic process. The tasks are arranged in a hierarchy; therefore, with the exception of the highest-level task (CONSULT), each task is a subtask of at least one other task (see Figure 8.2). It is not a simple hierarchy, since a task may be a subtask of a number of other tasks. For example, TEST-HYPOTHESIS is a subtask of GROUP-AND-DIFFERENTIATE, PURSUE-HYPOTHESIS, and findout. Furthermore, the hierarchy contains some recursive loops (e.g., through its subtasks, the task FORWARD-REASON may lead to invoking the task findout which, through its subtasks, could lead to the FORWARD-REASON task again). The subtasks of each task are fixed; FORWARD-REASON, for example, will always have clarify-finding, process-finding, and PROCESS-HYPOTHESIS as its three and only three subtasks.

Associated with each task is a group of metarules (Figure 8.3). Each metarule contains conditions that control whether or not the subtasks of this particular task can be tried.[3]

[3]Metarules carry out other actions besides calling subtasks, including much of the actual processing that gathers more information and manipulates the domain knowledge. However, to keep this description of NEOMYCIN's strategic knowledge and its use simple, these details will not be discussed here. Clancey & Bock, 1988, gives a more detailed description of how the tasks and metarules operate.

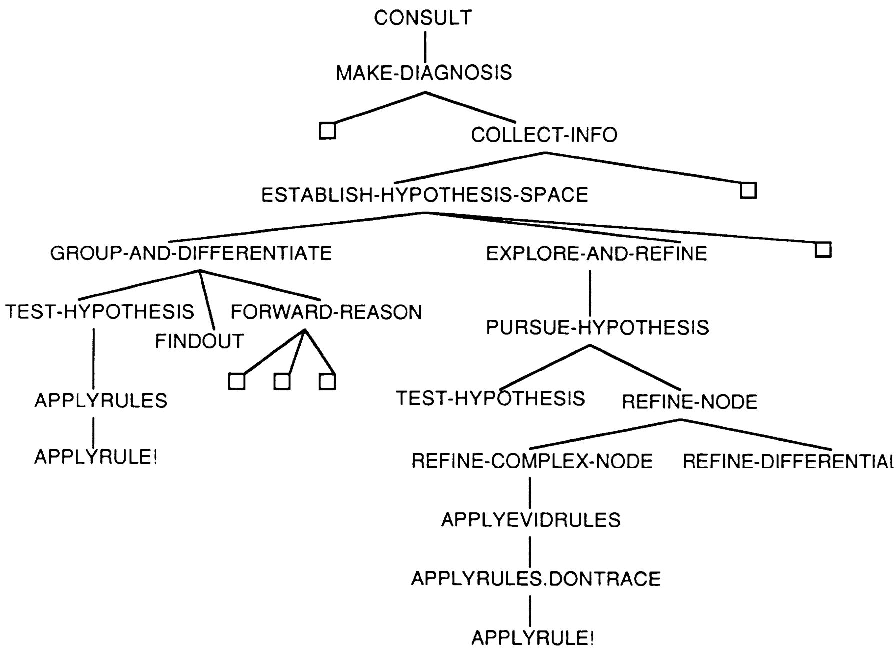

Figure 8.2: A simplified version of NEOMYCIN's Task Hierarchy. The boxes represent subtrees of tasks that have been omitted for the sake of readability.

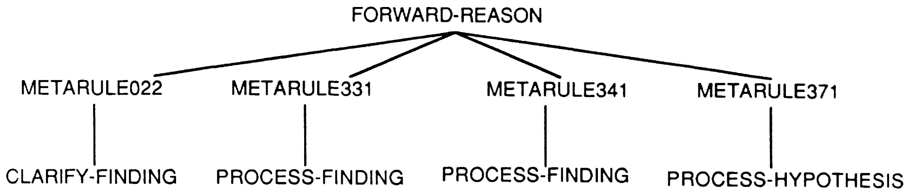

Figure 8.3: FORWARD-REASON and its associated metarules and subtasks

To run a diagnostic consultation, NEOMYCIN begins with the task at the top, CONSULT, and works its way in a procedural manner through the hierarchy. The task interpreter, which is the navigator through this procedural net, moves from tasks into their subtasks, then into their subtasks, sometimes iterating through a task a number of times until some condition (concerning the general state of the consultation) is met.

The portion of the consultation shown in Figure 8.1 was generated by NEOMYCIN. Figure 8.4 shows the path through the task/metarule hierarchy from the task CONSULT to the task FINDOUT of the focus STIFF-NECK-SIGNS, which generated Question 6 in that consultation.

NEOMYCIN, therefore, provides a way to express the strategy behind a sequence of diagnostic inquiries in terms of the task 'language.' This language is composed of all of NEOMYCIN's diagnostic tasks (some of which are shown in Figure 8.2) as well as the types of foci that these tasks use (such as *hypotheses*, *findings*, etc.). However, what part and how much of this hierarchy should be presented to students? Clearly, tasks that manipulate domain rules (e.g., APPLYRULE) are too embedded in the details of the implementation of NEOMYCIN to be of much use to the student. On the other hand, tasks such as "do a consultation" (CONSULT) or even "expand your differential" (ESTABLISH-HYPOTHESIS-SPACE) are very abstract, containing many distinct and significant subtasks. Introducing the student to just this level would blur too many important steps in diagnosis. Therefore a level of abstraction in between these levels has been chosen (see Figure 8.5).

CONSULT[ ]

RULE628

MAKE-DIAGNOSIS[ ]

RULE384

IDENTIFY-PROBLEM[ ]

RULE034

FORWARD-REASON[ ]

RULE341

PROCESS-FINDING[HEADACHE]

RULE148

APPLYRULES[RULE424]

RULE094

APPLYRULE![RULE424]

RULE095

FINDOUT[STIFF-NECK-ON-FLEXION]

RULE153

FINDOUT[STIFF-NECK-SIGNS]

RULE169

Figure 8.4: Tasks leading to Question 6 in Figure 8.1

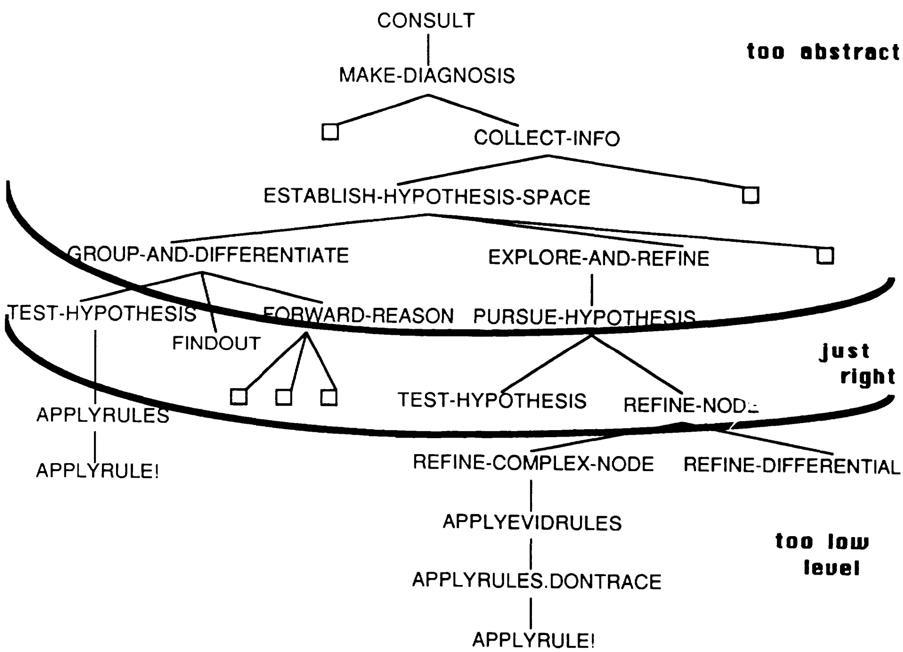

Figure 8.5: NEOMYCIN's Task Hierarchy divided to show the proper level for student actions

These tasks, such as TEST HYPOTHESIS, seem to offer the best balance of concreteness and relevance to the process of diagnosis.

Note that there are reasons to introduce the student to other parts of the hierarchy. For example, more general tasks such as ESTABLISH HYPOTHESIS SPACE group together tasks (such as TEST HYPOTHESIS) that we present to the student, and thus provide a more general language. However, for our purposes we focus on isolating the basic tasks or actions that can be used as building blocks for a structured diagnostic session.

It is important to note that there is considerable disagreement within the medical world as to what an appropriate strategy for diagnosis is. Leaper and colleagues found considerable variance in diagnostic 'procedure' among physicians and even between cases done by the same physician. Their results "suggested that any automated diagnostic system must be flexible enough to accommodate the wishes of a variety of clinicians" (Leaper et al., 1973). Thus flexibility was very important in the design of GUIDON-MANAGE. Two medical students and two physicians were consulted during the process of selecting the appropriate tasks to present to medical students. The student is not confined to NEOMYCIN's overall strategy, only to the language of tasks. As was mentioned earlier, the NEOMYCIN task hierarchy was developed with an expert teacher/clinician. While his reasoning strategy is captured in these tasks (Clancey & Letsinger, 1984), the approach relates well to other projects in medical reasoning (e.g., Rubin, 1975; Elstein, Shulman & Sprafka, 1978; Kassirer & Gorry, 1978; Kassirer, Kuipers & Gorry, 1982).

8.2 A GUIDON-MANAGE Session

This section leads the reader through part of a GUIDON-MANAGE session from a student's perspective. The next section will examine the processing below the surface.

The student has received before this point a brief introduction to the use and purpose of the system. Originally this was done orally and informally by the experimenter/observer; however, we now have a short, on-line introduction to the system that describes the mode of interaction and the tools available to the student (Barnhouse, 1988). Recall that the goal of GUIDON-MANAGE is to acquaint the students with a language of tasks that will help them structure and articulate a strategy for diagnosis. As the mode of interaction, the student performs a diagnosis (or consultation) by selecting tasks that the system will execute. Requests for patient data are indirectly generated by the system during execution of any task and are answered from a stored patient data record.

8.2.1 Interacting with the System

The student enters GUIDON-MANAGE by selecting the option "START GUIDON-MANAGE" from the options under the pulldown menu CONSULT. The student is then prompted to choose a patient case from a short list. In the current example the student has chosen SUZANNE, CASE 1002. A short paragraph

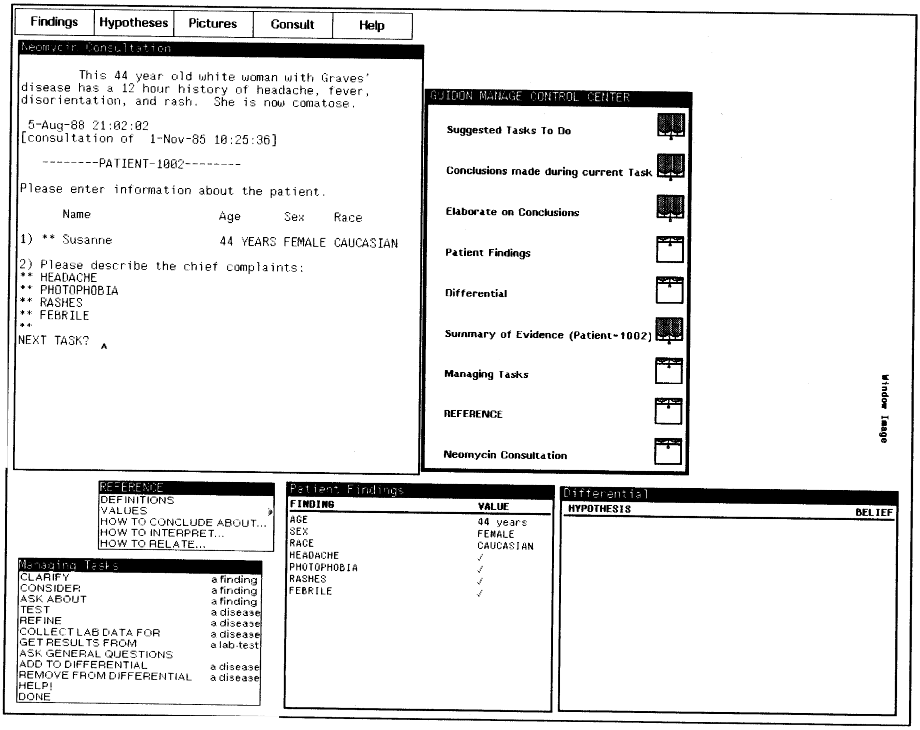

Figure 8.6: Initial view of GUIDON-MANAGE session

describing the patient and presenting symptoms (i.e., the chief complaints that brought the patient to the doctor) is then displayed:

> This 44 year old white woman with Graves' disease has a 12 hour history of headache, fever, disorientation, and rash. She is now comatose.

Figure 8.6 shows what the screen now looks like; the chief complaints have been gathered, and the system is waiting for the student to select a task to execute.

The student now chooses a task from the *Managing Tasks Menu.* In this case the student opts to clarify the finding *febrile* (the fact that the patient has a fever). As the *Consultation/Typescript Window* in Figure 8.7 shows, this task causes the system to ask about the patient's temperature.

```
Neomycin Consultation (Q3)

27-May-87 20:46:50
[consultation of  1-Nov-85 10:25:36]

   --------PATIENT-1002--------

Please enter information about the patient.

      Name                     Age      Sex    Race

1) ** Susanne                  44 YEARS FEMALE CAUCASIAN

2) Please describe the chief complaints:
** HEADACHE
** PHOTOPHOBIA
** RASHES
** FEBRILE
**
NEXT TASK?  CLARIFY-A-FINDING

Please enter the FINDING name: FEBRILE

3) What is Susanne's temperature (in Fahrenheit)?
** 105.8F
NEXT TASK?
```

Figure 8.7: *Consultation/Typescript Window* after first task/focus pair is chosen

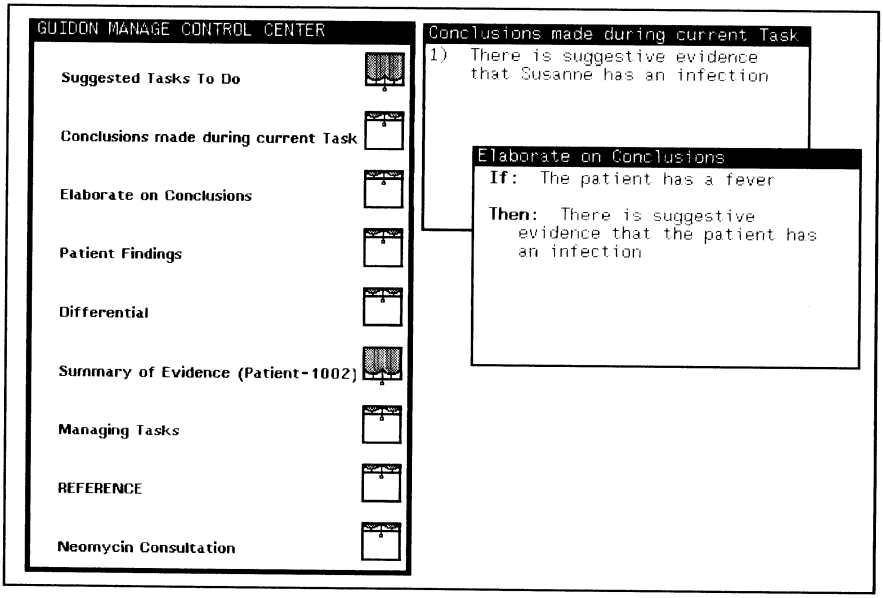

Figure 8.8: Conclusion and elaboration

Next, the student decides to consider the fever (i.e., the finding *febrile*). An hourglass icon appears in the *Control Center* by the window title for the *Conclusions Window*, indicating that a conclusion has been made. Selecting this icon updates the *Conclusions Window*, indicating that there is suggestive evidence that the patient has an infection. The differential window is updated as well to reflect this new conclusion, that is, that NEOMYCIN has evidence for an infectious process. The student then selects the text of this conclusion and generates the elaboration shown in Figure 8.8.

The student continues executing tasks in this manner. More questions are asked, conclusions are made, and abstractions of data are formed (for example, *headache chronicity* is determined from *headache duration*). However, suppose that the student is now stuck and needs a suggestion about what to do next. The task HELP! is selected, and the *Suggested Tasks To Do Window* is opened, displaying the suggestion to TEST the hypothesis *meningitis* (Figure 8.9). The student may then choose this task and its associated focus, closing the *Suggested Tasks To Do Window*, and continue with the consultation.

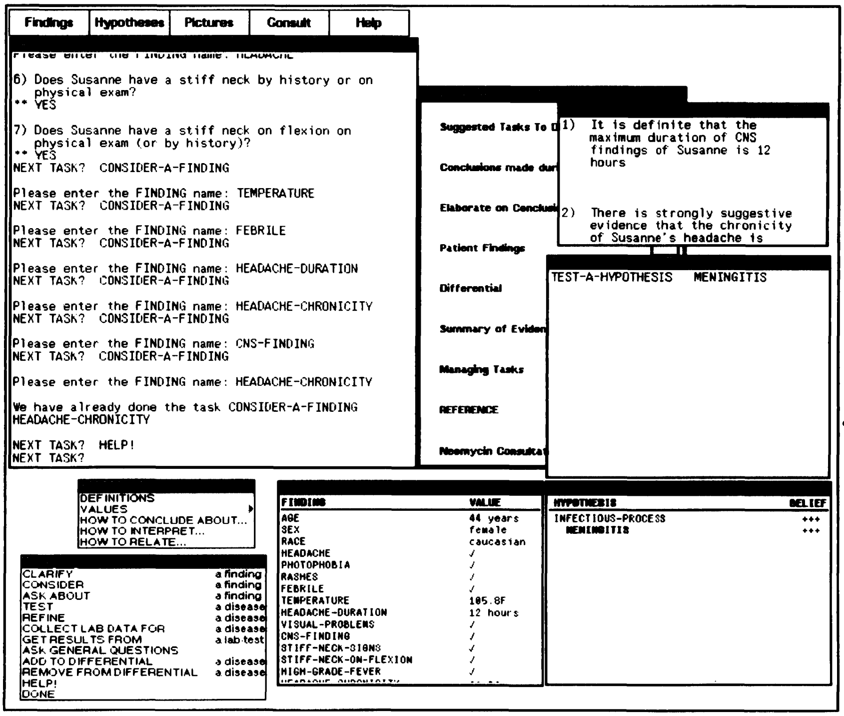

Figure 8.9: Asking for HELP! within GUIDON-MANAGE

This brief tour is not a complete session, but should be enough to give the reader a feeling for the style of interaction between the student and the system. The task language itself is the only strong constraint imposed on the user. The student is able to explore how any task works and when that task should be done. There are also a number of capabilities for examining the medical knowledge, both static (independent of any consultation, e.g., all possible findings associated with meningitis) and dynamic (the facts for this particular case). These capabilities (in addition to those in the *Reference Menu*) are part of GUIDON-WATCH (Richer & Clancey, 1985).

8.2.2 The Interface

As illustrated above, the student examines and selects items from a series of windows and menus that comprise the user interface. The following sections describe in more details what the student can do.

Choosing a Task

A student may select a task by buttoning one of the tasks in the *Managing Tasks Menu* or the *Suggested Tasks To Do Window* or by typing his or her desired task directly into the *Consultation/Typescript Window* at the *Next Task* prompt. If the task requires a focus (such as TEST HYPOTHESIS, which needs a specific hypothesis as the focus of its testing), a prompt will be printed in the *Consultation Window* for the appropriate type of focus (e.g., "Please enter the finding name," or "Please enter the HYPOTHESIS name," or "Please enter the LAB TEST name"). All questions and their responses generated during a session are printed in the *Consultation Window*.

The *Managing Tasks Menu* contains all of the tasks that the student can choose to execute. When a student clicks the mouse button on one of these tasks, this task is printed in the *Consultation/Typescript Window* along with a prompt for a focus where appropriate. To get a brief description of what a particular task does, the student may hold the mouse button down over that task.

By selecting a task, the student is indicating that he or she would like to perform a particular action or operation. These operations are often performed upon a particular focus. The convention used here (borrowed from the ALGEBRALAND system (Brown, 1985)) displays the task in upper-case letters and its focus type—either *disease* (hypothesis), *finding*, or *lab* *test*—in lowercase letters. Below are brief descriptions of each of these tasks.

CLARIFY a finding gathers more information about the given piece of data. For example, when clarifying the finding *headache* one might ask about the headache's duration and severity.

CONSIDER a finding looks for relationships between the given findings and categories of hypotheses or findings.

ASK ABOUT a finding directly inquires whether or not the patient has the particular finding. The student is discouraged from using this task, because it doesn't force him or her to make the justification for the question explicit.

TEST a disease tries to rule in or rule out the given hypothesis.

REFINE a disease expands the list of hypotheses on the differential to include the subtypes of the given hypothesis.

COLLECT LAB DATA FOR a disease requests the results from lab tests which would help rule in or rule out the given hypothesis. These results must then be CONSIDER-ed for implications to be noted and for the differential to be updated. This task is analogous to TEST *a hypothesis*. However, because lab data is dealt with separately in medicine, a distinct task is supplied.

GET RESULTS FROM a lab test queries the patient database for all the results from a particular lab test. For example, if one chooses *Complete Blood Count* (CBC) then NEOMYCIN generates questions regarding the *White Blood Cell* count (WBC), PMNS, BANDS, etc. This task is similar to a sequence of ASK ABOUT tasks for all results associated with the given lab test.

ADD TO DIFFERENTIAL a disease puts the given hypothesis in the *Differential* and *Summary of Evidence* windows.

REMOVE FROM DIFFERENTIAL a disease removes the given hypothesis from these two windows.

HELP! provides suggestions of task(s) the student could choose next (See the section on "Getting Assistance," below).

DONE prompts the user for confirmation and then ends the session.

Storing Data and Hypotheses

During a consultation, information is gathered about various findings associated with the patient as well as hypotheses of the patient's malady. The *Patient Findings Window* keeps track of all the patient data that the student has already collected. It is updated every time the student learns about a new piece of data.

The *Differential* is a list of the current hypotheses for which NEOMYCIN or the student have some evidence. Associated with each hypothesis is the degree of belief in that hypothesis (on a scale from +3 to −3). Both of these windows can be used to supply the focus of a task in the *Consultation/Typescript Window* or to bring up a menu of more information about the selected hypothesis (e.g., what findings are evidence for this hypothesis) or finding (e.g., what diseases can cause this finding).

An alternate means of viewing the differential is also provided, called the *Summary of Evidence*. This window lists the findings that have been used to provide evidence for each hypothesis on the differential.

Following the Reasoning Process

Each time the system makes a conclusion (e.g., *There is suggestive evidence that Suzanne has meningitis*), the conclusion is printed in the *Conclusions Window*. The most recent conclusion is always at the top and numbered one (*1*). The student may select, using the mouse, the text of any conclusion to get an elaboration of that conclusion, that is, a justification of why a conclusion was made. The window is also scrollable to allow the student to examine the conclusions made at any point in the session. Conclusions' elaborations are printed in a separate, scrollable window.

Getting Assistance

The *Suggested Tasks To Do Window* is opened when the student asks for help. It displays one or more task/focus pairs (e.g., TEST-A-HYPOTHESIS *meningitis*) as suggestions for the student to try. When the student simply selects a task and focus as the next set to be executed, they are entered in the *Consultation/Typescript Window* at the NEXT TASK? prompt. The student has the option to take the suggestion or not; however, the window remains open until one of the suggested tasks is chosen. A reference menu (Figure 8.10) has also been provided to offer explanations of medical terms

Figure 8.10: Reference menu of medical terms

and their use, such as how various findings and diseases are related and the normal values for given lab tests.

Control Center

In response to the students' need for more structure in the interface, a *Control Center* was provided (Figure 8.11). The name of each window or menu in the system appears in this control window. When an item is selected, the program indicates what the selected window is currently displaying, how it is updated, and how it can be used by the student. It will also flash the window if it is open. Next to each window name appears an icon of a window with a blind either open or drawn, depending on whether the corresponding window is open or closed. This icon changes dynamically as the state of the window changes, and can be selected itself to change the window's state. Finally, if a window is waiting for the student (e.g., the *Conclusions Window* is ready to show the student a new conclusion, but is waiting until the student is ready to see it), a small hourglass icon will appear in the *Control Center* next to the window's name. Selecting this hourglass resumes processing within the waiting window.

The students' response to this window has been overwhelmingly positive. It gives the student a focus in a (possible) sea of windows. The hourglasses are particularly useful, enabling the student easily to see when and where the program is waiting.

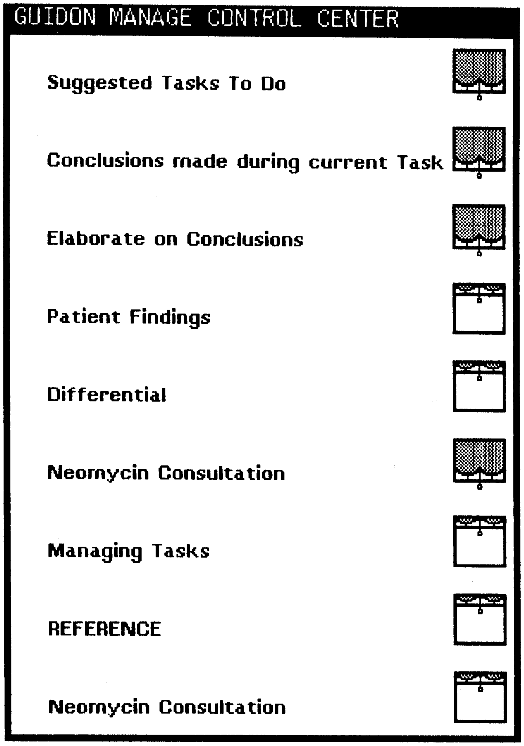

Figure 8.11: *Control Center Window*

8.3 Design and Implementation Issues

8.3.1 Major Design Considerations

GUIDON-MANAGE is part of a larger project to develop a series of tutoring systems that will combine to give a student a comprehensive introduction to the process of diagnosis. The focus of the GUIDON-MANAGE program is the language of diagnosis. Creating an environment to teach the language of an active process required two major design decisions.

First, the language itself had to be formulated. In our research, this was a three-step process. The first step was the knowledge acquisition performed by Clancey and Letsinger (1984) as they developed the task language for the NEOMYCIN system. The second source of information was classes in the Stanford Medical School that teach the diagnostic process through role-playing exercises. Observations here helped identify the basic language already familiar to students (enabling the analysis summarized by Figure 8.5). Finally, suggestions and criticisms were solicited from both the medical staff affiliated with the project and other medical students. The final result of this process is the menu of tasks (Figure 8.6).

The second design consideration was that each of the tasks must generate some visible action in the consultation environment. This allows the student to see exactly what each task does. For example, testing a hypothesis involves aggregating data that lends evidence towards that particular hypothesis. The conclusion and elaboration windows demonstrate the process of evidence gathering that occurs during this task. New evidence is also used to update the differential window. We believe that for the system to hold the student's interest continually, it must clearly react to the student's commands. This allows the student to feel direct responsibility for the actions that he or she requests the system to carry out.

The sections that follow describe the implementation of these design considerations.

8.3.2 Division of NEOMYCIN's Task Hierarchy into Levels of Abstraction

Beneath the surface of the interface to the student, there exists another kind of interface—the one between NEOMYCIN's task interpreter and GUIDON-

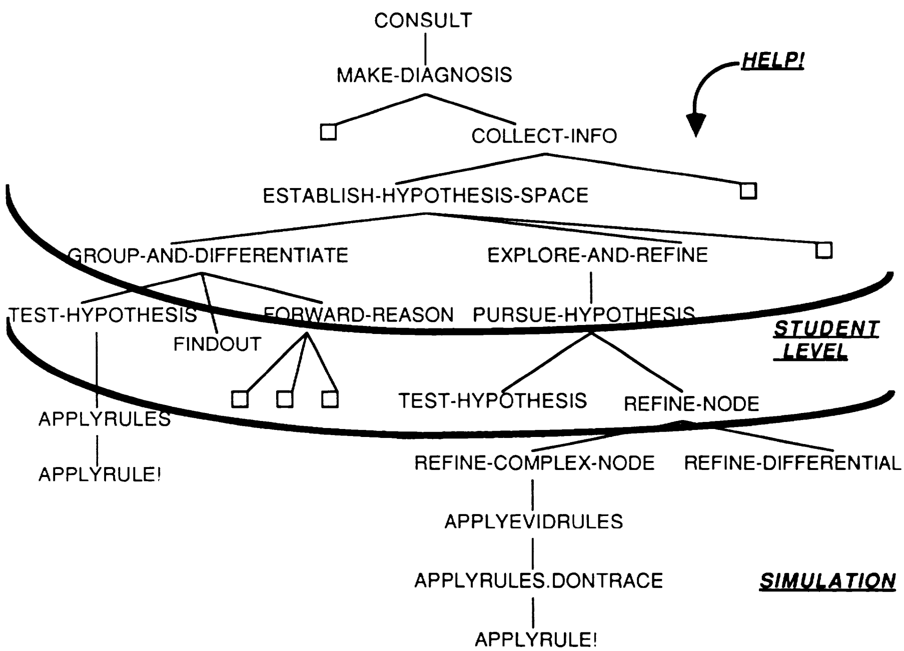

Figure 8.12: Levels of abstraction in Task Hierarchy

MANAGE's task interpreter. In Section 8.1, the notion of levels of abstraction within the NEOMYCIN task hierarchy was introduced. This section will discuss how these different levels are used within GUIDON-MANAGE.

Recall that NEOMYCIN's task hierarchy can be broken into three levels of abstraction (Figure 8.12); the tasks at each level are composed of tasks from the level below them.

Middle-Level Tasks—Student. The middle level consists of the tasks that have the appropriate balance of generality and specificity to be relevant to the student (Section 8.1). With one exception, which will be discussed later, only the student can decide that a task at this level will be executed. A trapping mechanism prevents the system from carrying out these tasks.

Low-Level Tasks—Simulation. The low-level tasks are used for simulation. When a student tells the system to execute a middle-level task, its metarules are applied, and consequently the lower-level tasks, which are subtasks of the student's selected task, are executed. Within these primitive tasks GUIDON-MANAGE monitors for interesting changes in the environment, such as questions being generated, conclusions being made, or new data being induced, which will be shown to the student.

High-Level Tasks—Assistance. The HELP! mechanism uses the high-level tasks to generate suggestions to the student for possible task/focus pairs to try next. The GUIDON-MANAGE task interpreter instructs NEOMYCIN's task interpreter to work through the task hierarchy from the *high*-level tasks until NEOMYCIN tries to execute a student level task (with a particular focus). This task/focus pair is then suggested to the student. Because these high-level tasks take into account the current state of the consultation—what data is known, what hypotheses are on the differential, etc.—this method of generating suggestions is actually telling the student what NEOMYCIN would do next in the current situation.

8.3.3 Lookahead

The system also has the capability to prune irrelevant suggestions before they are shown to the student. Whenever the HELP! mechanism is about to post a suggestion, it does a quick check of the conditions of all the metarules associated with the task to be suggested. If all of these metarules will fail (i.e., not fire because their conditions are not met), then the task/focus pair is not suggested to the student, but instead is carried out by the system, for bookkeeping purposes. This prevents the program from suggesting a task that will have no effect. This is the one exception, mentioned earlier, in which the system applies a student-level task.

8.3.4 Completeness and Flexibility

Two of the most vital issues in the design and implementation of GUIDON-MANAGE are completeness and flexibility. Completeness, in this context, means that if a student executes the tasks suggested by the system (or at least the most significant ones), the final differential will correlate highly with one generated by NEOMYCIN running the same case outside of a tutoring session.

Recall that this is important because the student's strategy should be allowed to vary within certain bounds from NEOMYCIN's approach. A different ordering from gathering patient data should not greatly alter the final diagnosis. The most difficult hurdle to overcome in ensuring completeness is to determine what it means to carry out a task. Once these parameters are established for the student-level tasks, then enough processing must be carried out to cover these parameters without overstepping the boundaries of the particular task being executed. For example, if CONSIDER-ing a finding *X* causes the conclusion to be made that there is evidence for hypothesis *Y*, then *Y* should be added to the differential as part of the CONSIDER task. However, NEOMYCIN would normally then proceed to process and test *Y*. This should not be done as part of the CONSIDER task, but rather should be interrupted until the student decides to do this processing himself.

The *flexibility* criterion tests the modularity of the tasks within the hierarchy. The system needs the capability to run any (student or higher-level) task at any time. This means that the task interpreter will be denied a predictable context within which to execute a task. There will be no guarantee of what came before or after. Two key methods used for sustaining flexibility are (1) to prevent the system from marking tasks as completed when they were only partially finished and (2) to simulate some of these bookkeeping notations when we want the system to skip over a task (e.g., if the system tries to initiate the execution of student-level tasks during its processing).

Empirical trials show that both completeness and flexibility are achieved within the system. The student can easily reproduce NEOMYCIN's final differential even with significant variations to the overall strategy and ordering of the tasks.

8.3.5 More Implementation Details

Finally, let us look at a session from inside the system. After a patient case is chosen by the user, the system enters the GUIDON-MANAGE module. The windows comprising the interface are initialized, hooked into the control center mechanism, and set into their 'beginning of consultation' state (i.e., open or closed). GUIDON-MANAGE invokes the task interpreter with the top-level NEOMYCIN task, CONSULT, and allows the system to continue processing until it traps on a student-level task. The system is now properly initialized, the *Initial Data* has been shown to the user, and usually a list has been generated of suggested tasks through FORWARD-REASON.[4] This method of using NEOMYCIN's tasks instead of creating special ones for GUIDON-MANAGE gives flexibility to future system builders to make changes to the task hierarchy. (The approach is somewhat inelegant, however, because processing must 'pop out' of these initial tasks before their subtasks are completed, leaving them in limbo. The tasks CONSULT and MAKE-DIAGNOSIS, for example, are technically never completed.)

At this point we enter into a loop in which GUIDON-MANAGE is waiting for the student to input a task, exiting when DONE has been chosen and confirmed. When a student selects a task and focus, processing continues until the task is completed or the system tries to execute one of the student-level tasks. If the system traps on a student-level task, the *lookahead* mechanism tests to see if any of the metarules of this task will succeed, possibly causing some subtask to be tried and producing some effect on the session (e.g., it might make a conclusion). If not, then the system simply carries out this task (for bookkeeping purposes) and continues processing. If any one of the metarules does succeed, the task is added to the suggested tasks and a flag is set to indicate that the system will be popping out of the task interpreter. This flag is important to prevent side effects that would occur when a task has actually been completed. Extra processing must also be prevented after the system pops up each level. Control then returns again to the main loop in the GUIDON-MANAGE module, and the system is once again waiting for user input.

The flowchart in Figure 8.13 reviews the interactions between the student, GUIDON-MANAGE, and NEOMYCIN, as discussed previously.

[4]Note that a mechanism within the GUIDON-MANAGE interpreter allows FORWARD-REASON to continue iterating until it has exhausted its possibilities (most of which are added to the suggestions list). FORWARD-REASON is the only one of the higher-level tasks that is permitted to iterate within GUIDON-MANAGE, because it corresponds to immediate, automatic associations.

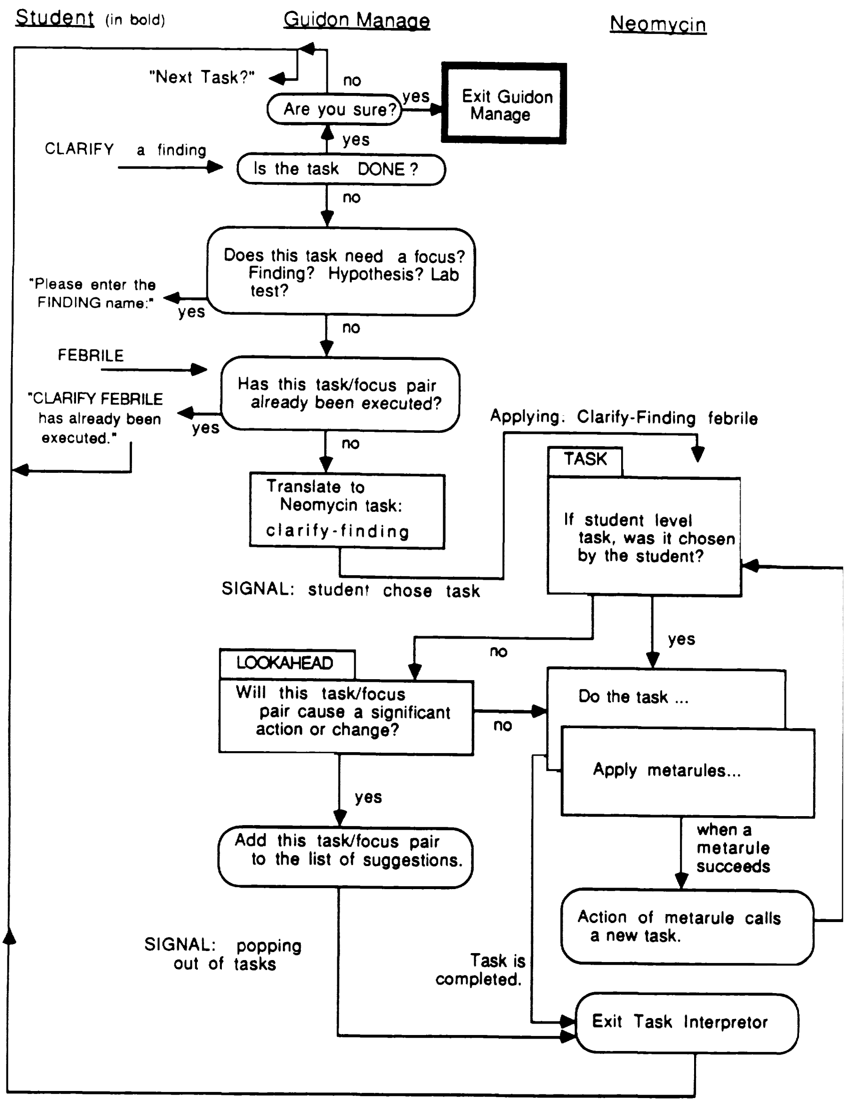

Figure 8.13: Interactions between student, GUIDON-MANAGE, and NEOMYCIN

8.4 Student Trials

Five second- and third-year medical students from the Stanford University Medical School[5] participated in trials of GUIDON-MANAGE. One of these students experimented with a much earlier version of the system in a more informal trial providing feedback that was critical to the changes reflected in the current version of the system. The official trials lasted from one to two hours, and the students were paid for their time. The purpose of these trials was to gather feedback from the students and observe their interactions with the system. Formal data analysis was not attempted, for several reasons: (1) the group was too small to be able to draw any statistically significant conclusions; (2) the students varied greatly in their prior experience with diagnosis; (3) the students were just becoming familiar with the system by the end of the session (multiple sessions might be interesting); and (4) the system itself varied from session to session since changes based on feedback from one session were incorporated into the next session.

The students were introduced to the system by either an oral or on-line presentation that described the tasks, the tools, and the purpose of the system. All of the students had had some previous experience with computer interfaces and the mouse, so only a brief review was given of the mechanics of using the system. The students were encouraged to be curious and critical of the system. Sessions were tape recorded and students were asked to vocalize their thoughts, frustrations, questions, and criticisms continuously.

8.4.1 Observations

Students were overwhelmed, especially at first, by the complexity of the system. It was difficult for them to keep track of all the tools that were available. They had trouble figuring out exactly what was expected of them when the system paused.

Paradoxically, the students wanted the power of all those tools and the information that they offered. Almost every question that they had about the case could be answered by using one of these tools. How *would* I conclude *meningitis*? Why was this conclusion made? What does this term mean? This situation was greatly improved with the inclusion of the

[5] These students were in their second or third year out of five years.

Control Center Window (Figure 8.11). Much less assistance was required of the observer after this window had been added.

All of the students found that they understood the language of diagnostic tasks only after experimenting with them. This was, after all, the designed purpose of the program. Tasks such as *Clarify a finding* and *Consider a finding* were particularly difficult because they were similar enough to be confused. They look at the same piece of information in two different ways and both tasks must be executed to complete processing of a finding. More generally, directionality of tasks was a problem at the start of almost all the sessions. Directionality refers to whether the task aggregates data or analyzes its constituents. For example, *Clarify a finding* gathers more information about the given finding by asking about its constituent findings, (e.g., *duration* of a headache may be considered a constituent finding of the finding *headache*). In contrast, *Consider a finding* views the finding as something to be explained or aggregated with other data.

For the most part, the students liked the exploratory nature of the system. One student especially enjoyed the cooperative aspect of the environment in which she was in control of the expert's action. None of the students had any trouble getting accustomed to the mode of interaction of choosing a task for the system to execute. However, the students became frustrated when they selected a task and nothing happened. This was caused at times by deficiencies in NEOMYCIN's medical knowledge, but more usually by the irrelevance of the task. This problem inspired the lookahead mechanism. Now, for the most part, the tutor does not suggest something that has no effect on the consultation environment.

There was a high variability in the general reactions of the students. One student, who had had no clinical or preclinical experience and who did not know the medical information within the system's knowledge base very well, felt that the system was quite useful. She told the observers that she would definitely want to run through a few sessions before facing a resident quizzing her on what to do next. However, another student who had recently begun her clinical training felt that the system held her back. This student had an excellent grasp of the medical knowledge and was annoyed by the system's deficiencies. At one point she said, "I don't think this was as useful to me since I already had a preconceived idea of how to do diagnosis." The more advanced students became frustrated with the explicitness of the task language, feeling that the various tasks were too basic. This was an expected reaction, since experts in many fields feel bogged down when they are forced to resort to a step-by-step process of problem solving (Gentner & Stevens, 1983).

Some sort of critiquing mechanism would have been helpful, at least for the novices. These students, especially, tend to wander. Such exploring may be very useful since the students did eventually see that they were getting nowhere, but they themselves stated that they would have liked a bit more guidance from the system. However, the more expert student did not use an overall strategy that paralleled NEOMYCIN's strategy. Any critique or constraint based on the system's strategy might have exacerbated her frustration with the system, unless the program clearly demonstrated the value of its approach. An interesting observation is that all of the students wanted to know lab test results fairly early in the consultation, whereas NEOMYCIN and many teachers would require these tasks to be done much later. In every case, there was important history and physical information that had not yet been uncovered. Consider the following exchange between the observer and a student, for example:

> *The student has found out that the patient has a headache and a stiff neck on flexion. The system concludes that there is a possibility that the patient has meningitis. The student immediately chooses to gather lab data results associated with meningitis.*
>
> *Observer*: Would you always go right to lab data at this point?
>
> *Student*: If there is a possibility of meningitis? Yes. It's the only way to test it out.

The student does not find out until later that the patient had a fever (which among other things can be an indication of meningitis). A physician was later asked whether this student was right in asking for lab data so soon in the consultation. He responded that there was no way that the student should be ordering a lumbar puncture (one of the lab tests ordered) without knowing, at the very least, whether the patient had a fever or not. Providing feedback of this type requires presenting important constraints among the tasks.

8.5 Conclusion

The GUIDON-MANAGE project was established to create an environment to introduce medical students to the process of diagnosis. This was achieved by introducing a language of diagnostic tasks and an environment in which to experiment with these tasks. We have demonstrated the capability to run the strategic tasks in an independent, flexible and complete manner. The concept of partitioning tasks into a three-level hierarchy has also proven very useful. This final section discusses the lessons learned from this project.

As with any user-directed system, the interface has played an important role in the relative success of GUIDON-MANAGE. The idea of *direct manipulation* (Hutchins, Hollan & Norman, 1986), where the user can manipulate objects on the screen and feel as if he or she, and not the system, is in control, could be carried even further than it currently is in the system. For example, mechanisms for moving and closing windows could be made simpler. The *Control Center* definitely handled much of the confusion about what was going on during a session, but plans are underway to make the interface simpler yet. The *Conclusions Window*, for instance, will be combined with the *Typescript Window*, giving the student a view of the conclusions in their proper context.

It is interesting to look at how some of Collins, Brown, and Newman's six methods of teaching came into play in our system. The students were introduced to a part of the expert's model of diagnosis through the task language. Scaffolding was used to help the students run a consultation with the expert carrying out the tasks selected by the student and dealing with most of the medical domain issues. The students were allowed a large degree of independence in exploring the environment, which was both good and bad. The students learned a great deal about both medical and diagnostic knowledge by using a "What if I tried to...?" strategy. However, more coaching is needed within this environment. One possibility is to have students watch NEOMYCIN solve a problem, then put them into a GUIDON-MANAGE session, and finish with a mixed-initiative discussion about the same problem. This would allow the students to reflect on what they have done and to compare it to what the program does.

A student modeler program could provide a basis for feedback. Two prototype modeling systems are now ready for use in GUIDON-MANAGE

(Wilkins, Clancey & Buchanan, 1985; London, 1986). However, with so much variability even among the experts, tolerance for deviation from NEOMYCIN is important. As mentioned before, the high-level metarules within the hierarchy exercise certain constraints on the overall ordering of tasks (e.g., *History and Physical* always precedes *Lab Data* collection). Annotating the significant orderings for teaching is fairly straightforward (see Clancey, 1984, for further discussion).

Of particular interest are trials to determine how learning the language of tasks helps the student articulate impasses and difficult problems, as well as explain to him- or herself how a peer or teacher is approaching a problem. Hence, we could further explore our claim that an instructional program facilitates learning in general by facilitating a dialogue about the process of reasoning itself.

Acknowledgements

The authors wish to thank Bruce G. Buchanan and the GUIDON2/NEOQ-MYCIN group for all their support. They also wish to thank Bevan Yueh, a medical student, for developing the reference menu. This work was supported by grants from the Josiah Macy, Jr. Foundation and the Office of Naval Research (Grant Number N00014-85-K-0305). Computational resources were provided by a grant from the Sumex-Aim National Resource (NIH Grant RR00785).

References

Anderson, J.R., Boyle, C.F. & Yost, G. (1985). The geometry tutor. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI), Volume 1*, Los Angeles, CA, 1–7.

Anderson, J.R., Farrell, R.G. & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science*, *8*(2), 87–129.

Barnhouse, S. (1988). GUIDON-TOURS. *Proceedings of the Intelligent Tutoring Systems Conference*, Montreal, Quebec, Canada.

Brown, J.S. (1982). Learning by doing revisited for electronic learning environments. In M.A. White (ed.), *The Future of Electronic Learning*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Brown, J.S. (1985). Process versus product—A perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research, 1*, 179–201.

Brown, J.S., Burton, R.R. & De Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman & J.S. Brown (eds.), *Intelligent Tutoring Systems*, London, UK: Academic Press, 227–282.

Buchanan, B.G. & Shortliffe, E.H. (1984). *Rule-Based Expert Systems: The* MYCIN *Experiments of the Stanford Heuristic Programming Project.* Reading, MA: Addison-Wesley.

Burton, R.R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J.S. Brown (eds.), *Intelligent Tutoring Systems*, London, UK: Academic Press, 157–183.

Carbonell, J.R. & Collins, A. (1973). Natural semantics in artificial intelligence. *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI)*, 344–351.

Clancey, W.J. (1984). *Acquiring, Representing, and Evaluating a Competence Model of Diagnostic Strategy.* HPP Memo 84-2, Knowledge Systems Laboratory, Stanford University, Stanford, CA. To appear in M.T.H. Chi, R. Glaser & M. Farr (eds.), *The Nature of Expertise*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Clancey, W.J. (1986). From GUIDON to NEOMYCIN and HERACLES in twenty short lessons. (ONR Final Report 1979–1985). *AI Magazine, 7*(3), 40–60.

Clancey, W.J. (1987). Intelligent tutoring systems: A tutorial survey. In van Lamsweerde (ed.), *International Professorship Series: 1985*, London, UK: Academic Press.

Clancey, W.J. & Bock C. (1988). Representing control knowledge as abstract tasks and metarules. In Bolc & Coombs (eds.), *Computer Expert Systems.*

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

Clancey, W.J., Richer, M.H., Wilkins, D.C., Barnhouse, S., Kapsner, C., Leserman, D., Macias, J., Merchant, A. & Rodolitz, N. (1986). GUIDON-DEBUG*: The Student as Knowledge Engineer.* KSL Working Paper No. 86-34, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Collins, A. & Brown, J.S. (in press). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (eds.), *Learning Issues for Intelligent Tutoring Systems*, New York, NY: Springer Verlag.

Collins, A., Brown, J.S. & Newman S.E. (1986). *Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics.* BBN Technical Report 6459, Bolt Beranek and Newman, Inc., Cambridge, MA. (November.)

Elstein, A.S., Shulman, L.S. & Sprafka, S.A. (1978). *Medical Problem Solving: An Analysis of Clinical Reasoning.* Cambridge, MA: Harvard University Press.

Gentner, D. & Stevens, A. (eds.), (1983). *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Harless, W.G., Drennon, G.G., Marxer, J.J., Root, J.A. & Miller, G.E. (1971). CASE: A computer-aided simulation of the clinical encounter. *Journal of Medical Education, 46*, 443–448.

Hollan, J.D., Hutchins, E.L. & Weitzman, L. (1984). STEAMER: An interactive inspectable simulation-based training system. *The AI Magazine, 5*(2), 15–27.

Hutchins, E.L., Hollan, J.D. & Norman, D.A. (1986). Direct manipulation interfaces. In D.A. Norman & S.W. Draper (eds.), *User Centered System Designs: New Perspectives on Human-Computer Interactions*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Johnson, W.L. & Soloway, E.M. (1985). PROUST: An automatic debugger for PASCAL programs. *BYTE Magazine, 10*, 179–190.

Kassirer, J.P. & Gorry, G.A. (1978). Clinical problem solving: A behavioral analysis. *Annals of Internal Medicine, 89*, 245–255.

Kassirer, J.P., Kuipers, B.J. & Gorry, G.A. (1982). Toward a theory of clinical expertise. *The American Journal of Medicine, 73*, 251–259.

Leaper, D.J., Gill, P.W., Staniland, J.R., Horrocks, J.C. & de Dombal, F.T. (1973). Clinical diagnostic process: An analysis. *British Medical Journal, 3*, 569–574.

London, B. (1986). *Diagnostic Student Modelling with Multiple Viewpoints by Plan Inference.* Thesis proposal, Department of Computer Science, Stanford University, Stanford, CA.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* New York, NY: Basic Books, Inc.

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

Rubin, A.D. (1975). *Hypothesis Formation and Evaluation in Medical Diagnosis.* Technical Report AI-TR-316, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Sleeman, D. & Brown, J.S. (1982). *Intelligent Tutoring Systems.* London, UK: Academic Press.

VanLehn, K. (1983). Human procedural skill acquisition: Theory, model, and psychological validation. *Proceedings of the National Conference on AI*, Washington, D.C., August, 420–423.

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

Young, R.M. (1983). Surrogates and mappings: Two kinds of conceptual models for interactive devices. In D. Gentner & L.A. Stevens (eds.), *Mental Models*, Hillsdale, NJ: Lawrence Erlbaum Associates.