9

Interactive Transfer of Expertise

Randall Davis

Whereas much early work in artificial intelligence was devoted to the search for a single, powerful, domain-independent problem-solving methodology [e.g., GPS (Newell and Simon, 1972)], subsequent efforts have stressed the use of large stores of domain-specific knowledge as a basis for high performance. The knowledge base for this sort of program [e.g., DENDRAL (Feigenbaum et al., 1971), MACSYMA (Moses, 1971)] is often assembled by hand, an ongoing task that may involve several person-years of effort. A key element in constructing a knowledge base is the transfer of expertise from a human expert to the program. Since the domain expert often knows nothing about programming, the interaction between the expert and the performance program usually requires the mediation of a human programmer.

We have sought to create a program that could supply much the same sort of assistance as that provided by the programmer in this transfer of expertise. The result is a system called TEIRESIAS[1] (Davis, 1976; 1978; Davis et al., 1977), a large Interlisp program designed to offer assistance in the interactive transfer of knowledge from a human expert to the knowledge base of a high-performance program (Figure 9-1). Information flow from right to left is labeled *explanation.* This is the process by which TEIRESIAS clarifies for the expert the source of the performance program's results and motivations for its actions. This is a prerequisite to knowledge acquisition, since the expert must first discover what the performance pro-

This chapter originally appeared in *Artificial Intelligence* 12: 121–157 (1979). It has been shortened and edited.

[1]The program is named for the blind seer in *Oedipus the King,* since the program, like the prophet, has a form of "higher-order" knowledge.

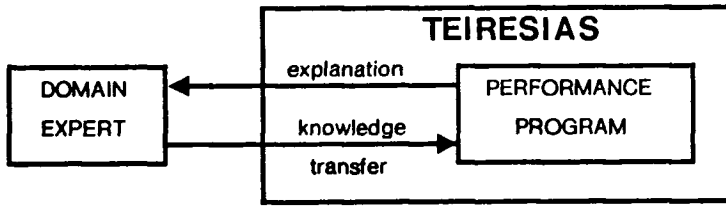


FIGURE 9-1 Interaction between the expert and the performance program is facilitated by TEIRESIAS.

gram already knows and how it used that knowledge. Information flow from left to right is labeled *knowledge transfer.* This is the process by which the expert adds to or modifies the store of domain-specific knowledge in the performance program.

Work on TEIRESIAS has had two general goals. We have attempted first to develop a set of tools for knowledge base construction and maintenance and to abstract from them a methodology applicable to a range of systems. The second, more general goal has been the development of an intelligent assistant. This task involves confronting many of the traditional problems of AI and has resulted in the exploration of a number of solutions, reviewed below.

This chapter describes a number of the key ideas in the development of TEIRESIAS and discusses their implementation in the context of a specific task (acquisition of new inference rules[2]) for a specific rule-based performance program. While the discussion deals with a specific task, system, and knowledge representation, several of the main ideas are applicable to more general issues concerning the creation of intelligent programs.

9.1 Meta-Level Knowledge

A central theme that runs through this chapter (and is discussed more fully in Part Nine) is the concept of *meta-level knowledge,* or knowledge about knowledge. This takes several different forms, but can be summed up generally by saying that a program can "know what it knows." That is, not only can a program use its knowledge directly, but it may also be able to examine it, abstract it, reason about it, and direct its application.

To see in general terms how this might be accomplished, recall that

[2]Acquisition of new conceptual primitives from which rules are built is discussed by Davis (1978), while the design and implementation of the explanation capability suggested in Figure 9-1 is discussed in Part Six.

one of the principal problems of AI is the question of representation of knowledge about the world, for which numerous techniques have been developed. One way to view what we have done is to imagine turning this in on itself, using some of these same techniques to describe the program itself. The resulting system contains both *object-level* representations, which describe the external world, and *meta-level* representations, which describe the internal world of representations. As the discussion of rule models in Sections 9.6 and 9.7 will make clear, such a system has a number of interesting capabilities.

9.2 Perspective on Knowledge Acquisition

We view the interaction between the domain expert and the performance program as *interactive transfer of expertise.* We see it in terms of a teacher who continually challenges a student with new problems to solve and carefully observes the student's performance. The teacher may interrupt to request a justification of some particular step the student has taken in solving the problem or may challenge the final result. This process may uncover a fault in the student's knowledge of the subject (the debugging phase) and result in the transfer of information to correct it (the knowledge acquisition phase). Other approaches to knowledge acquisition can be compared to this by considering their relative positions along two dimensions: (i) the sophistication of their debugging facilities, and (ii) the independence of their knowledge acquisition mechanism.

The *simplest* sort of debugging tool is characterized by programs like DDT, used to debug assembly language programs. The tool is totally passive (in the sense that it operates only in response to user commands), is low-level (since it operates at the level of machine or assembly language), and knows nothing about the application domain of the program. Debuggers like BAIL (Reiser, 1975) and Interlisp's break package (Teitelman, 1974) are a step up from this since they function at the level of programming languages such as SAIL and Interlisp. The explanation capabilities in TEIRESIAS, in particular the HOW and WHY commands (see Part Six for examples), represent another step, since they function at the level of the control structure of the application program. The guided debugging that TEIRESIAS can also provide (illustrated in Section 9.5) represents yet another step, since here the debugger is taking the initiative and has enough built-in knowledge about the control structure that it can track down the error. Finally, at the most sophisticated level are knowledge-rich debuggers like the one described by Brown and Burton (1978). Here the program is active, high-level, informed about the application domain, and capable of independently localizing and characterizing bugs.

By *independence* of the knowledge acquisition mechanism, we mean the degree of human cooperation necessary. Much work on knowledge acqui-

sition has emphasized a highly autonomous mode of operation. There is, for example, a large body of work aimed at inducing the appropriate generalizations from a set of test data; see, for example, Buchanan and Mitchell (1978) and Hayes-Roth and McDermott (1977). In these efforts user interaction is limited to presenting the program with the data and perhaps providing a brief description of the domain in the form of values for a few key parameters; the program then functions independently. Winston's work on concept formation (Winston, 1970) relied somewhat more heavily on user interaction. There the teacher was responsible for providing an appropriate sequence of examples (and nonexamples) of a concept. In describing our work, we have used the phrase "interactive transfer of expertise" to indicate that we view knowledge acquisition as information transfer from an expert to a program. TEIRESIAS does not attempt to derive new knowledge on its own, but rather tries to "listen" as attentively as possible, commenting appropriately to help the expert augment the knowledge base. It thus requires strong cooperation from the expert.

There is an important assumption involved in the attempt to establish this sort of communication: we are assuming that it is possible to distinguish between the *problem-solving paradigm* and the *expertise* or, equivalently, that control structure and representation in the performance program can be considered separately from the content of its knowledge base. The basic control structure(s) and representations are assumed to be established and debugged, and the fundamental approach to the problem is assumed to be acceptable. The question of *how* knowledge is to be encoded and used is settled by the selection of one or more of the available representations and control structures. The expert's task is to enlarge *what* it is the program knows.

There is a corollary assumption, too, in the belief that the control structures and knowledge representations can be made sufficiently comprehensible to the expert that he or she can (a) understand the system's behavior in terms of them and (b) use them to codify his or her own knowledge. This ensures that the expert understands system performance well enough to know what to correct, and can then express the required knowledge, i.e., can "think" in those terms. Thus part of the task of establishing the link shown in Figure 9-1 involves insulating the expert from the details of implementation, by establishing a discourse at a level high enough that he or she does not have to program in LISP.

9.3 Design of the Performance Program

Figure 9-2 shows the major elements of the performance program that TEIRESIAS is designed to help construct. Although the performance program described here is MYCIN, the context within which TEIRESIAS was

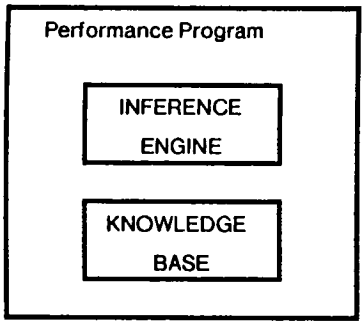


FIGURE 9-2 Architecture of the performance program.

actually developed, many of the features of TEIRESIAS have been incorporated in EMYCIN (see Chapter 15) and are independent of any domain. The *knowledge base* is the program's store of task-specific knowledge that makes possible high performance. The *inference engine* is an interpreter that uses the knowledge base to solve the problem at hand. The main point of interest in this very simple design is the explicit division between these two parts of the program. This design is in keeping with the assumption noted above that the expert's task is to augment the knowledge base of a program whose control structure (inference engine) is assumed to be both appropriate and debugged.

Two important advantages accrue from keeping this division as strict as possible. First, if all of the control structure information has been kept in the inference engine, then we can engage the domain expert in a discussion of the knowledge base alone rather than of questions of programming and control structures. Second, if all of the task-specific knowledge has been kept in the knowledge base, then it is possible to remove the current knowledge base, "plug in" another, and obtain a performance program for a new task (see Part Five). The explicit division thus offers a degree of domain-independence. It does not mean, however, that the inference engine and knowledge base are totally independent: knowledge base content is strongly influenced by the control paradigm used in the inference engine. It is this unavoidable interaction that motivates the important assumption, noted in Section 9.2, that the control structure and knowledge representation are comprehensible to the expert, at least at the conceptual level.

An example of the program in action is shown in Section 9.5. The program interviews the user, requesting various pieces of information that are relevant to selecting the most appropriate antibiotic therapy, then prints its recommendations. In the remainder of this chapter the user will

be an expert running MYCIN in order to challenge it, offering it a difficult case and observing and correcting its performance.

We have noted earlier that the expert must have at least a high-level understanding of the operation of the inference engine and the manner of knowledge representation in order to be able to express new knowledge for the performance program. An example of a rule, with brief explanations of the terms premise, Boolean combination, conclusion, and certainty factor, suffices to allow understanding of the representation of knowledge. An equally brief explanation of backward chaining and the conservative strategy of exhaustive evidence gathering suffices to allow understanding of the inference engine. As mentioned in Section 9.2, we are assuming that the expert can understand these concepts without having to deal with details of implementation. Note as well that TEIRESIAS' basic design and the notion of interactive transfer of expertise do not depend on this particular control structure, only on the (nontrivial) assumption that an equally comprehensible explanation can be found for whatever control structure is actually used in the inference engine.

9.4 A Word About Natural Language

Natural language has not been a major focus of this work, and for the most part we have used the simplest techniques that would support the level of performance required. For instance, all questions and responses from TEIRESIAS are either preformed or manufactured by filling in templates with appropriate words (as evidenced by the occasional appearance of ungrammatical phrases like "a organism"). Where answers are free text (rather than multiple choice), the system relies primarily on the keyword- and template-oriented techniques described in Chapter 18. There is no parser in the system, in part to minimize processing time and in part because users often give ungrammatical responses.

9.5 Example of TEIRESIAS in Operation

In the interaction shown in Figure 9-3, the expert is challenging MYCIN with a particular case, observing the program's performance, and then making the necessary correction to the knowledge base. The bug encountered here was created by deleting an existing rule from the knowledge base and removing all references to it. The expert then repairs the bug by tracking down the source of the problem and teaching the system the "new"

rule. The deleted rule was chosen in order to display a large number of TEIRESIAS's capabilities in a single example. It is special in no other sense, however, and offers a real trace of the system in action.

In the trace, computer output is in mixed upper and lower case, user responses follow a double asterisk and are in bold upper case, and commentary is indented, italicized, and enclosed in parentheses. The example is also divided into five phases for later reference.

(MYCIN has been provided with details of a case and has just concluded likely causes of the infection.)

My therapy recommendation will be based on the following probable infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA
<Item 1> E.COLI [ORGANISM-1]
<Item 2> PSEUDOMONAS-AERUGINOSA [ORGANISM-1]

Are all these identities correct?
++** **N**

(The expert disagrees with the results and will indicate to TEIRESIAS the identities that he thinks should have resulted.)

(PHASE I: TRACKING DOWN THE BUG)

Are there any identities missing?
++** **Y**

Which one(s)?
++** **KLEBSIELLA-PNEUMONIAE PROTEUS-NON-MIRABILIS**

Are there any identities that shouldn't have appeared?
++** **PSEUDOMONAS-AERUGINOSA**

Do you want to
1] track down the problem yourself, or
2] allow me to guide the debugging, or
3] ignore it for now and fix it another time?

Type 1, 2 or 3
++** **2**

We'll start with the PSEUDOMONAS-AERUGINOSA first.

(This time, TEIRESIAS will guide the process. It applies the heuristic of starting with an identity that appeared incorrectly.)

The following rules were used in deducing that the identity of ORGANISM-1 is pseudomonas-aeruginosa.

RULE184

Since [1.1] the category of ORGANISM-1 is not known
[1.2] the gram stain of ORGANISM-1 is gramneg
[1.3] the morphology of ORGANISM-1 is rod

FIGURE 9-3 TEIRESIAS example.

[1.4] the aerobicity of ORGANISM-1 is facultative
There is weakly suggestive evidence (.3) that the identity of ORGANISM-1 is pseudomonas-aeruginosa

Is this rule correct?
+ + ** **Y**

Should its premise have failed for this case?
+ + ** **Y**

Which clause of the premise should have been false? [give #]
+ + ** **1**

No conclusions were reached concerning the category of ORGANISM-1
Should it have been possible to conclude a category?
+ + ** **Y**

Which one?
+ + ** **ENTEROBACTERIACEAE**

There were no rules successfully invoked in the attempt to deduce that the category of ORGANISM-1 is enterobacteriaceae.

The following (potentially applicable) rules were *not* executed because the starred [**] clause failed:

RULE116

IF:
1) the identity of ORGANISM-1 is not known
** 2) the gram stain of ORGANISM-1 is not known **
3) the morphology of ORGANISM-1 is not known
4) the site of CULTURE-1 is csf
5) the infection is meningitis
6) the age (in years) of the patient is less than or equal to .17

THEN: There is weakly suggestive evidence (.3) that the category of ORGANISM-1 is enterobacteriaceae

RULE050

IF:
1) the morphology of ORGANISM-1 is rod
2) the gram stain of ORGANISM-1 is gramneg
3) the aerobicity of ORGANISM-1 is facultative
** 4) the infection with ORGANISM-1 was acquired while the patient was hospitalized **

THEN: There is evidence that the category of ORGANISM-1 is enterobacteriaceae

Are all these rules correct?
+ + ** **Y**

Should any of them have been successfully invoked?
+ + ** **N**

(This would be the case if one of the starred premise clauses was false because it had been "stepped on" by yet another rule further back in the chain.)

Is there a rule missing?
+ + ** **Y**

FIGURE 9-3 continued

Ok, then I need a rule that would allow me to deduce that the category
of ORGANISM-1 is enterobacteriaceae.
Would you like to give me such a rule now?
+ + ** **Y**

Good . . .

(To review briefly: The expert is tracking down the reason why MYCIN incorrectly concluded that the identity was pseudomonas-aeruginosa. Pseudomonas-aeruginosa appeared because Rule 184 executed successfully. This occurred in part because category was unknown. Category was unknown because all rules to conclude it (50 and 116) failed. Since the expert believes all existing rules are correct, the knowledge base must lack a rule that will conclude about category; in particular, that the category is enterobacteriaceae. This will cause category to be known, inhibiting Rule 184, fixing at least this first problem, the incorrect conclusion about pseudomonas-aeruginosa.

Note how sharply focused the issue has become, for both the expert and TEIRESIAS, as a result of the large amount of contextual information available. The doctor is not being asked to "discuss bacteremia"; instead, the system can at this point say: "I need a rule that would allow me to deduce that the category of ORGANISM-1 is enterobacteriaceae." And it must be a rule which is invocable in the context of this patient. This focus makes it easier for the expert to specify a rule that may never have been explicitly stated or recognized as such. This can be an important advantage in any domain where the knowledge is not yet fully formalized.

The contextual information is useful for TEIRESIAS as well. It provides a number of expectations about the content of the rule that are useful both in understanding what the expert types in and in checking the new rule to make sure that it does indeed solve the problem at hand. Details of this process are explained below.

The trace continues with the expert typing in the rule . . .)

(PHASE II: INTERPRETING THE RULE)

The new rule will be called **RULE383**

If: 1 - **THE PATIENT'S INFECTION IS PRIMARY-BACTEREMIA**
and 2 - **THE SITE IS ONE OF THE STERILE SITES**
and 3 -
Then: 1 - **THERE IS EVIDENCE (.8) THAT THE CATEGORY IS ENTEROBACTERIACEAE**
and 2 -

This may take a few moments . . .

(Generally, it takes on the order of 10 to 30 seconds of cpu time.)

This is my understanding of your rule:

RULE383

IF: 1) Primary-bacteremia is an associated infection for which therapy would cover, and
2) The infection with the organism was acquired while the patient was hospitalized, and
3) The culture was taken from a sterile source, and
4) The organism is not normally found at this site

THEN: There is strongly suggestive evidence (.8) that the category of the organism is enterobacteriaceae

(PHASE III: MODIFYING THE INTERPRETATION)

(TEIRESIAS' initial interpretation of the natural language is incorrect (recall that the example was chosen in part for this reason). In the system's first attempt to understand the rule, each line of the original text has been interpreted as two clauses. This results primarily from the word-by-word approach to meaning. For instance, despite the obvious grammatical simplicity of the first line of text, the system used two connotations of the word "infection" to generate the first two clauses. In the second line of text the word "sterile" was responsible for clause 3, and the word "sites" used as the basis for clause 4. The details of this process are explained below, where it will become clear that while the translation is wrong, it is not unreasonable given the simple natural language facilities.

Nor is it unreasonable that TEIRESIAS has turned each line of English into more than one premise clause. The expert is not restricted to typing the English equivalent of a single premise clause on each line. If he were, rather than typing "the organism is a gram-negative aerobic rod," he would have to type three lines.

We will see later that while TEIRESIAS' interpretations are incorrect, they are all "reasonable" and "consistent," in light of the system's somewhat primitive approach to natural language. The rest of the example shows that even when the initial guesses are incorrect, straightening things out may be very easy.)

Okay? (Yes or No)
++** **N**

(The expert indicates that there is a problem with the interpretation and uses the system's rule editor to specify the changes needed.)

Change premise?
++** **Y**

What clause should be changed in premise?
(type HELP for list of commands)
++** **CHANGE 1**

Suppose (1) were changed to read-
1 - The infection is primary-bacteremia
Would that be ok?
++** **Y**

(Several simple heuristics are invoked to select the next most likely interpretation, based on the English source of this clause and the set of interpretations already rejected. In this case, the second choice is correct.)

Then you'll probably want to delete:

2) The infection with the organism was acquired while the patient was hospitalized

Correct?
++** **Y**

(TEIRESIAS enforces consistency constraints on its interpretation of the English text. One of these constraints indicates that if the new interpretation of clause 1 is correct, then clause 2 must have arisen spuriously. That is, the system realizes it cannot get both the new version of clause 1 and the old clause 2 from the same line of text, and indicates this to the expert.)

FIGURE 9-3 continued

Anything else?
++** **D 3**
clause(s) 3 deleted.

(Realizing that there was another spurious clause, the expert deletes clause 3, but then has second thoughts about the one he chose, so he checks, using the ? command.)

Anything else?
++** **?**
1 - the infection is primary-bacteremia
and 2 - <deleted>
and 3 - <deleted>
and 4 - the organism is not normally found at this site

(He then discovers that he has typed the wrong number, undoes the damage, and deletes the one initially intended. All commands can be abbreviated to a single letter.)

Anything else?
++** **U 3**
clause(s) 3 undeleted.

Anything else?
++** **D 4**
clause(s) 4 deleted.

(There is one last correction to be made for which TEIRESIAS suggests the wording . . .)

Anything else?
++** **C 3**
Suppose (3) were changed to read-
3 - The site of the culture is one of: those sites that are
normally sterile

Would that be ok?
++** **Y**

Anything else?
++** **?**
1 - the infection is primary-bacteremia
and 2 - <deleted>
and 3 - the site of the culture is one of: those sites that are
normally sterile
and 4 - <deleted>

++** **OK**

(. . . and at last he's happy with the rule.)

Change action?
++** **N**

This is my understanding of your rule:

RULE383
IF: 1) The infection is primary-bacteremia, and
2) The site of the culture is one of: those sites that are normally sterile
THEN: There is strongly suggestive evidence (.8) that the category of the organism is enterobacteriaceae

Okay? (Yes or No)
+ +** **Y**

(This is the classic test of comprehension: the original English text has been turned into a rule coded in LISP; the LISP is then translated back into English for approval. Compare the results with the expert's original input.

Now that the expert is satisfied that the system understands the rule as he has entered it, TEIRESIAS checks several more things about the rule.)

(PHASE IV: SECOND-GUESSING)

I hate to criticize, Dr. Davis, but did you know that most rules about what the category of a organism might be, that mention -

the site of a culture
the infection

Also mention -

[A] - the portal of entry of a organism

Shall I try to write a clause to account for [A]?
+ +** **Y**

(The mechanism that makes possible this sort of second-guessing by TEIRESIAS is explained in detail below. Briefly, the system maintains a model of the knowledge base. In checking to see how this new piece of information "fits into" that model, it discovers that its expectations have been only partially met. It indicates to the expert what expectations were left unfulfilled and asks if it should attempt to fill in the missing parts.)

How about-

[A] The portal of entry of the organism is GI

Ok?
+ +** **Y**

(The expert agreed to the inclusion of a clause about the portal of entry; this determines which attribute to use. TEIRESIAS examines its model of the knowledge base to see which predicate function is likely to be the right one; the model evidently suggested the function SAME. Finally, in order to fix the original bug, it must be possible to invoke the new rule for this patient. Since, in the original consultation, the expert responded that the portal of entry was the GI tract, this is a plausible guess for the value to use. This demonstrates another benefit of the "knowledge acquisition in context" methodology mentioned in Chapter 7.)

Also, most rules about what the category of a organism may be
Also conclude about -

[A] - the identity of a organism

Shall I try to write a clause to account for [A]?

FIGURE 9-3 continued

++** **N**

(The system noticed another unfulfilled expectation concerning the action part of the rule, but the expert always has the option of overriding the system's suggestions, and does so.)

The rule now reads:

RULE383

IF: 1) The infection is primary-bacteremia, and
2) The site of the culture is one of: those sites that are normally sterile, and
3) The portal of entry of the organism is GI

THEN: There is strongly suggestive evidence (.8) that the category of the organism is enterobacteriaceae

(Now that both the expert and TEIRESIAS are satisfied with the rule, there is a bit of bookkeeping to do. First, the new rule is checked by the system (behind the scenes) to be sure that there is nothing obviously wrong with it—for instance, one of the tests ensures that premise clauses are not contradictory. Had this been the case, the system would have noticed it at this point and the user would have been given the option of returning to the rule editor to fix the problem. There are several other things the system checks at this point, but this rule passes all the tests, and hence there is no indication of any problem.)

(PHASE V: FINAL CHECKOUT, BOOKKEEPING, RERUNNING THE CONSULTATION)

Please describe briefly why you added (or changed) this rule. Type an empty line when done.

++** **THE SYSTEM FAILED TO REALIZE IT COULD CONCLUDE**
++** **CATEGORY, AND THIS ALLOWED RULE184 TO INCORRECTLY**
++** **CONCLUDE IDENTITY**
++**

(Over the years of MYCIN development, the experts associated with the project have occasionally looked in astonishment at a rule which, though it had been around for some time, seemed, superficially at least, to make very little sense. The question then arises as to why that rule was added to the knowledge base. To avoid this problem, the expert is required to explain why he wrote or changed the rule.

There are also a number of other bookkeeping-type chores that are being attended to here, again, behind the scenes. TEIRESIAS takes care of tagging the rule with information that will simplify keeping track of the large, and constantly changing, body of rules. We will see in a moment how that information is accessed.)

RULE383 has now been added to the knowledge base.

I will now rerun the consultation to check the effectiveness of your new rule. Get comfortable, since this may take a bit.

Also, new questions may be asked as a result of this new rule. If a question is printed, please answer it with reference to this current patient.

(TEIRESIAS now invokes the performance program as a subprocess to make sure the bug is fixed, and to test for any other side effects. Since MYCIN has saved all the answers from the initial consultation, the

process is fairly inexpensive. The expert is warned about new questions because his new rule may mention an attribute that was not previously traced, or it may indirectly cause the tracing of one. While there are some cases in which it is possible to compute the incremental effect on a given consultation of adding a single new rule to the knowledge base, these are rare enough that in the long run it is computationally cheaper to simply rerun the consultation.)

18-DEC-75 16:30:46
----------PATIENT-281----------
---------INFECTION-1----------
----------CULTURE-1----------
----------ORGANISM-1----------

(The answers to all questions were available internally, so the performance program asks nothing new. The context names are printed to let the expert follow the progress of the consultation.)

My therapy recommendation will be based on the following probable
infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA
<Item 1> E. COLI [ORGANISM-1]
<Item 2> KLEBSIELLA-PNEUMONIAE [ORGANISM-1]
<Item 3> PROTEUS-NON-MIRABILIS [ORGANISM-1]

Well, it appears that things are straightened out.

(The new conclusions are the ones the expert indicated earlier as the correct answers. This does not go unnoticed by TEIRESIAS, which, as before, interrupts the consultation at this point to check the validity of the answers produced.)

Are all these values correct now?
+ + ** **Y**

(As always, TEIRESIAS gives the expert the last word, and waits for approval before proceeding.)

Good, then the problem's been fixed.
. . . continuing with therapy selection now.

(At this point TEIRESIAS allows the performance program subprocess to continue with the normal therapy-selection phase. This has been omitted here for brevity.

The reader may be wondering why three errors disappeared with the addition of one rule. Once the new rule permitted the category of the organism to be concluded, not only did Rule 184 fail, but a second rule that had previously failed was now successfully invoked. This rule used the category of the organism to conclude that klebsiella-pneumoniae and proteus-non-mirabilis were likely identities.

All of the bookkeeping that was done is accessible via the INFOrmation command added to the question-answering facility of the performance program. This gives the expert the background for any rule in the system.)

** INFO 383

was written by Dr. Davis

on December 18, 1975

FIGURE 9-3 continued

for the following reason:

THE SYSTEM FAILED TO REALIZE IT COULD CONCLUDE CATEGORY, AND THIS ALLOWED RULE184 TO INCORRECTLY CONCLUDE IDENTITY.

for patient [281]

[who was described as follows:
CLASSIC CASE OF GRAM NEGATIVE ROD INFECTION FOR A PATIENT WITH A NON-NOSOCOMIAL DISEASE]

FIGURE 9-3 continued

9.6 How It All Works

9.6.1 Overview of the Main Ideas

Before reviewing the trace in more detail, we describe the ideas that make possible the capabilities displayed. This subsection serves primarily to name and briefly sketch each in turn; the details are supplied in subsequent subsections reviewing the example. [See Davis (1976) for more details.]

Knowledge Acquisition in Context

Performance programs of the sort TEIRESIAS helps create will typically find their greatest utility in domains where there are no unifying laws on which to base algorithmic methods. In such domains there is instead a collection of informal knowledge based on accumulated experience. This means an expert specifying a new rule may be codifying a piece of knowledge that has never previously been isolated and expressed as such. Since this is difficult, anything that can be done to ease the task will prove very useful.

In response, we have emphasized knowledge acquisition in the context of a shortcoming in the knowledge base. To illustrate the utility of this approach, consider the difference between asking the expert:

What should I know about the patient?

and saying to him:

Here is an example in which you say the performance program made a mistake. Here is all the knowledge the program used, here are all the facts of the case, and here is how it reached its conclusions. Now,

what is it that you know and the system doesn't that allows you to avoid making that same mistake?

Note how much more focused the second question is and how much easier it is to answer.

Building Expectations

The focusing provided by the context is also an important aid to TEIRESIAS. In particular, it permits the system to build up a set of expectations concerning the knowledge to be acquired, facilitating knowledge transfer and making possible several useful features illustrated in the trace and described below.

Model-Based Understanding

Model-based understanding suggests that some aspects of understanding can be viewed as a process of matching: the entity to be understood is matched against a collection of prototypes, or models, and the most appropriate model is selected. This sets the framework in which further interpretation takes place. While this view is not new, TEIRESIAS employs a novel application of it, since the system has a model of the knowledge it is likely to be acquiring from the expert.

Giving a Program a Model of Its Own Knowledge

We will see that the combination of TEIRESIAS and the performance program amounts to a system that has a picture of its own knowledge. That is, it not only knows something about a particular domain but also in a primitive sense knows what it knows and employs that model of its knowledge in several ways.

Learning as a Process of Comparison

We do not view learning as simply the addition of information to an existing base of knowledge, but instead take it to include various forms of comparison of the new information with the old. This of course has its corollary in human behavior: a student will quickly point out discrepancies between newly taught material and his or her current stock of information. TEIRESIAS has a similar, though very primitive, capability: it compares new information supplied by the expert with the existing knowledge base, points out inconsistencies, and suggests possible remedies.

Learning by Experience

One of the long-recognized potential weaknesses of any model-based system is dependence on a fixed set of models, since the scope of the program's "understanding" of the world is constrained by the number and types of models it has. As will become clear, the models TEIRESIAS employs are not handcrafted and static, but are instead formed and continually revised as a by-product of its experience in interacting with the expert.

9.6.2 Phase I: Tracking Down the Bug

To provide the debugging facility shown in the dialogue of Section 9.5, TEIRESIAS maintains a detailed record of the actions of the performance program during the consultation and then interprets this record on the basis of an exhaustive analysis of the performance program's control structure. This presents the expert with a comprehensible task because (a) the backward-chaining technique used by the performance program is straightforward and intuitive, even to a nonprogrammer, and (b) the rules are designed to encode knowledge at a reasonably high conceptual level. As a result, even though TEIRESIAS is running through an exhaustive case analysis of the preceding consultation, the expert is presented with a task of debugging *reasoning* rather than *code*.

The availability of an algorithmic debugging process is also an important factor in encouraging the expert to be as precise as possible in making responses. Note that at each point in tracking down the error the expert must either approve of the rules invoked and the conclusions made or indicate which one was in error and supply the correction. This approach is extremely useful in domains where knowledge has not yet been formalized and where the traditional reductionist approach of dissecting reasoning down to observational primitives is not yet well established.[3]

TEIRESIAS further encourages precise comments by keeping the debugging process sharply focused. For instance, when it became clear that there was a problem with the inability to deduce the category, the system first asked which category it should have been. It then displayed only those rules appropriate to that answer, rather than all the rules concerning that topic that were tried.

Finally, consider the extensive amount of contextual information that is now available. The expert has been presented with a detailed example

[3]The debugging process does allow the expert to indicate that the performance program's results are incorrect, but he or she cannot find an error in the reasoning. This choice is offered only as a last resort and is intended to deal with situations where there may be a bug in the underlying control structure of the performance program (contrary to our assumption in Section 9.2).

of the performance program in action, has available all of the facts of the case, and has seen how the relevant knowledge has been applied. This makes it much easier for him or her to specify the particular chunk of knowledge that may be missing. This contextual information will prove very useful for TEIRESIAS as well. It is clear, for instance, what the *effect* of invoking the new rule must be (as TEIRESIAS indicates, it must be a rule that will deduce that the category should be *Enterobacteriaceae*), and it is also clear what the *circumstances* of its invocation must be (the rule must be invocable for the case under consideration, or it won't repair the bug). Both of these pieces of information are especially useful in Phase II and Phase V.

9.6.3 Phase II: Interpreting the Rule

As is traditional, "understanding" the expert's natural language version of the rule is viewed in terms of converting it to an internal representation and then retranslating that into English for the expert's approval. In this case the internal representation is the Interlisp form of the rule, so the process is also a simple type of code generation.

There were a number of reasons for rejecting a standard natural language understanding approach to this problem. First, as noted, understanding natural language is well known to be a difficult problem and was not a central focus of this research. Second, our experience suggested that experts frequently sacrifice precise grammar in favor of the compactness available in the technical language of the domain. As a result, approaches that were strongly grammar-based might not fare well. Finally, technical language often contains a fairly high percentage of unambiguous words, so a simpler approach that includes reliance on keyword analysis has a good chance of performing adequately.

As will become clear, our approach to analyzing the expert's new rule is based on both simple keyword spotting and predictions TEIRESIAS is able to make about the likely content of the rule. Code generation is accomplished via a form of template completion that is similar in some respects to template completion processes that have been used in generating natural language. Details of all these processes are given below.

Models and Model-Based Understanding

To set the stage for reviewing the details of the interpretation process, we digress for a moment to consider the idea of models and model-based understanding, and then to explore their application in TEIRESIAS. In the most general terms, a *model* can be seen as a *compact, high-level description of structure, organization, or content* that may be used both *to provide a framework for lower-level processing* and *to express expectations about the world.* One

early, particularly graphic example of this idea can be found in the work on computer vision by Falk (1970). The task there was understanding block-world scenes; the goal was to determine the identity, location, and orientation of each block in a scene containing one or more blocks selected from a known set of possibilities. The key element of this work of interest to us here is the use of a set of *prototypes* for the blocks, prototypes that resembled wire frame models. Although such a description oversimplifies, part of the operation of Falk's system can be described in terms of two phases. The system first performed a preliminary pass to detect possible edge points in the scene and attempted to fit a block model to each collection of edges. The model chosen was then used in the second phase as a guide to further processing. If, for instance, the model accounted for all but one of the lines in a region, this suggested that the extra line might be spurious. If the model fit well except for some line missing from the scene, that was a good hint that a line had been overlooked and indicated as well where to go looking for it.

We can imagine one further refinement in the interpretation process, though it was not a part of Falk's system, and explain it in these same terms. Imagine that the system had available some *a priori* hints about what blocks might be found in the next scene. One way to express those hints would be to bias the matching process. That is, in the attempt to match a model against the data, the system might (depending on the strength of the hint) try the indicated models first, make a greater attempt to effect a match with one of them, or even restrict the set of possibilities to just those contained in the hint.

Note that in this system (i) the models supply a compact, high-level description of structure (the structure of each block), (ii) the description is used to guide lower-level processing (processing of the array of digitized intensity values), (iii) expectations can be expressed by a biasing or restriction on the set of models used, and (iv) "understanding" is viewed in terms of a matching and selection process (matching models against the data and selecting one that fits).

Rule Models

Now, recall our original task of interpreting the expert's natural language version of the rule, and view it in the terms described above. As in the computer vision example, there is a signal to be processed (the text), it is noisy (words can be ambiguous), and there is context available (from the debugging process) that can supply some hints about the likely content of the signal. To complete the analogy, we need a model that can (a) capture the structure, organization, or content of the expert's reasoning, (b) guide the interpretation process, and (c) express expectations about the likely content of the new rule.

Where might we get such a thing? There are interesting regularities

EXAMPLES—the subset of rules this model describes
DESCRIPTION—characterization of a typical member of this subset
- characterization of the premise
- characterization of the action

MORE GENERAL—pointers to models describing more general subsets of rules
MORE SPECIFIC—pointers to models describing more specific subsets of rules

FIGURE 9-4 Rule model structure.

in the knowledge base that might supply what we need. Not surprisingly, rules about a single topic tend to have characteristics in common—there are ways of reasoning about a given topic. From these regularities we have constructed *rule models.* These are abstract descriptions of subsets of rules, built from empirical generalizations about those rules and used to characterize a typical member of the subset.

Rule models are composed of four parts as shown in Figure 9-4. They contain, first, a list of EXAMPLES, the subset of rules from which this model was constructed. Next, a DESCRIPTION characterizes a typical member of the subset. Since we are dealing in this case with rules composed of premise-action pairs, the DESCRIPTION currently implemented contains individual characterizations of a typical premise and a typical action. Then, since the current representation scheme used in those rules is based on associative triples, we have chosen to implement those characterizations by indicating (a) which attributes typically appear in the premise (or action) of a rule in this subset and (b) correlations of attributes appearing in the premise (or action).[4] Note that the central idea is the concept of *characterizing a typical member of the subset.* Naturally, that characterization will look different for subsets of rules, procedures, theorems, or any other representation. But the main idea of characterization is widely applicable and not restricted to any particular representational formalism.

The two remaining parts of the rule model are pointers to models describing more general and more specific subsets of rules. The set of models is organized into a number of tree structures, each of the general form shown in Figure 9-5. At the root of each tree is the model made from all the rules that conclude about the attribute (i.e., the CATEGORY model), below this are two models dealing with all affirmative and all negative rules (e.g., the CATEGORY-IS model). Below this are models dealing with rules that affirm or deny specific values of the attribute. These models are not handcrafted by the expert. They are instead assembled by TEIRESIAS on the basis of the current contents of the knowledge base, in what amounts to a simple statistical form of concept formation. The combination of TEIRESIAS and the performance program thus presents a system that has a model of its own knowledge, one it forms itself.

[4]Both (a) and (b) are constructed via simple thresholding operations.

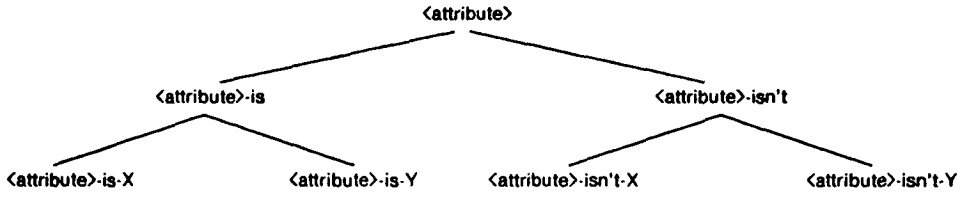


FIGURE 9-5 Organization of the rule models.

The rule models are the primary example of meta-level knowledge used in knowledge aquisition (for discussion of other forms, see Chapter 28). This form of knowledge and its generation by the system itself have several interesting implications illustrated in later sections.

Figure 9-6 shows a rule model; this is the one used by TEIRESIAS in the interaction shown earlier. (Since not all of the details of implementation are relevant here, this discussion will omit some.) As indicated above, there is a list of the rules from which this model was constructed, descriptions characterizing the premise and the action, and pointers to more specific and more general models. Each characterization in the description is shown

```
CATEGORY-IS

EXAMPLES    ((RULE116 .33)
             (RULE050 .78)
             (RULE037 .80)
             (RULE095 .90)
             (RULE152 1.0)
             (RULE140 1.0))

PREMISE     ((GRAM SAME NOTSAME 3.83)
             (MORPH SAME NOTSAME 3.83)
             ((GRAM SAME) (MORPH SAME) 3.83)
             ((MORPH SAME) (GRAM SAME) 3.83)
             ((AIR SAME) (NOSOCOMIAL NOTSAME SAME) (MORPH SAME)
                         (GRAM SAME) 1.50)
             ((NOSOCOMIAL NOTSAME SAME) (AIR SAME) (MORPH SAME)
                         (GRAM SAME) 1.50)
             ((INFECTION SAME) (SITE MEMBF SAME) 1.23)
             ((SITE MEMBF SAME) (INFECTION SAME) (PORTAL SAME)
                         1.23))

ACTION      ((CATEGORY CONCLUDE 4.73)
             (IDENT CONCLUDE 4.05)
             ((CATEGORY CONCLUDE) (IDENT CONCLUDE) 4.73))

MORE-GENL   (CATEGORY-MOD)

MORE-SPEC   NIL
```

FIGURE 9-6 Rule model for rules concluding affirmatively about CATEGORY.

split into its two parts, one concerning the presence of individual attributes and the other describing correlations. The first item in the premise description, for instance, indicates that most rules reaching conclusions about the category mention the attribute GRAM (for gram stain) in their premises; when they do mention it, they typically use the predicate functions SAME and NOTSAME; and the "strength," or reliability, of this piece of advice is 3.83 [see Davis (1976) for precise definitions of the quoted terms].

Correlations are shown as several lists of attribute-predicate pairs. The fourth item in the premise description, for example, indicates that when the attribute gram stain (GRAM) appears in the premise of a rule in this subset, the attribute morphology (MORPH) typically appears as well. As before, the predicate functions are those frequently associated with the attributes, and the number is an indication of reliability.

Choosing a Model

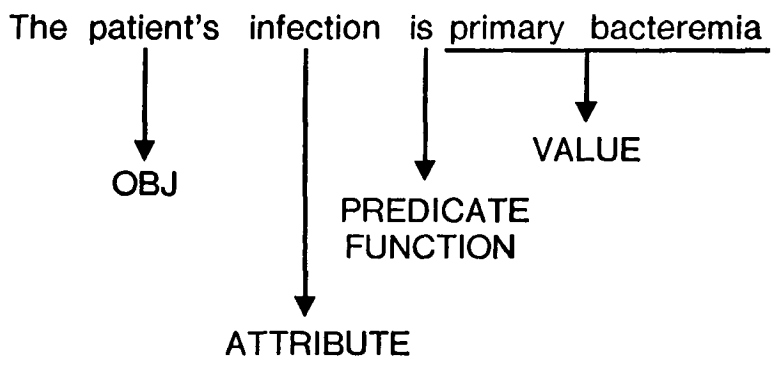
It was noted earlier that tracking down the bug in the knowledge base provides useful context and, among other things, serves to set up TEIRESIAS's expectations about the sort of rule it is about to receive. As suggested, these expectations are expressed by restricting the set of models that will be considered for use in guiding the interpretation. At this point TEIRESIAS chooses a model that expresses what it knows thus far about the kind of rule to expect, and in the current example it expects a rule that will deduce that the category should be *Enterobacteriaceae.*

Since there is not necessarily a rule model for every characterization, the system chooses the closest one. This is done by starting at the top of the tree of models and descending until either reaching a model of the desired type or encountering a leaf of the tree. In this case the process descends to the second level (the CATEGORY-IS model), notices that there is no model for CATEGORY-IS-ENTEROBACTERIACEAE at the next level, and settles for the former.[5]

Using the Rule Model: Guiding the Natural Language Interpretation

TEIRESIAS uses the rule models in two different ways in the acquisition process. The first is as a guide in understanding the text typed by the expert, as is described here. The second is as a means of allowing TEI-

[5]This technique is used in several places throughout the knowledge transfer process, and in general supplies the model that best matches the current requirements, by accommodating varying levels of specificity in the stated expectations. If, for instance, the system had known only that it expected a rule that concluded about category, it would have selected the first node in the model tree without further search. TEIRESIAS also has techniques for checking that the appropriate model has been chosen and can advise the expert if a discrepancy appears. See Davis (1976) for an example.



(a) Connotations found in the new rule.

Function	*Template*
SAME	(OBJ ATTRIBUTE VALUE)

(b) Template for the predicate function SAME.

1) (SAME CNTXT TREAT-ALSO PRIMARY-BACTEREMIA)
"Primary bacteremia is an associated infection for which therapy should cover."

2) (SAME CNTXT INFECTION PRIMARY-BACTEREMIA)
"The infection is primary bacteremia."

(c) Two choices for the resulting code (with translations).

FIGURE 9-7 Use of rule models to guide the understanding of a new rule.

RESIAS to see whether the new rule "fits into" its current model of the knowledge base in Phase IV.

To see how the rule models are used to guide the interpretation of the text of the new rule in the example, consider the first line of text typed by the expert in the new rule, Rule 383 (THE PATIENT'S INFECTION IS PRIMARY-BACTEREMIA). Each word is first reduced to a canonical form by a process that can recognize plural endings and that has access to a dictionary of synonyms (see Chapter 18). We then consider the possible connotations that each word may have (Figure 9-7a). Here connotation means the word might be referring to one or more of the conceptual primitives from which rules are built (i.e., it might refer to a predicate function, attribute, object, or value). One set of connotations is shown.[6]

Code generation is accomplished via a fill-in-the-blank mechanism. Associated with each predicate function is a *template* (see Chapter 5), a list structure that resembles a simplified procedure declaration and gives the

[6]The connotations of a word are determined by a number of pointers associated with it, which are in turn derived from the English phrases associated with each of the primitives.

order and generic type of each argument to a call of that function (Figure 9-7b). Associated with each of the primitives that make up a template (e.g., ATTRIBUTE, VALUE) is a procedure capable of scanning the list of connotations to find an item of the appropriate type to fill in that blank. The whole process is begun by checking the list of connotations for the predicate function implicated most strongly (in this case, SAME), retrieving the template for that function, and allowing it to scan the connotations and "fill itself in" using the procedures associated with the primitives. The set of connotations in Figure 9-7a produces the LISP code in Figure 9-7c. The ATTRIBUTE routine finds two choices for the attribute name, TREAT-ALSO and INFECTION, based on associations of the word infection with the phrases used to mention those attributes. The VALUE routine finds an appropriate value (PRIMARY-BACTEREMIA), the OBJect routine finds the corresponding object type (PATIENT) (but following the convention noted earlier, returns the variable name CNTXT to be used in the actual code).

There are several points to note here. First, the first interpretation in Figure 9-7c is incorrect (the system has been misled by the use of the word infection in the English phrase associated with TREAT-ALSO); we'll see in a moment how it is corrected. Second, several plausible (syntactically valid) interpretations are usually available from each line of text, and TEIRESIAS generates all of them. Each is assigned a score (the *text score*) indicating how likely it is, based on how strongly it was implicated by the text. Finally, we have not yet used the rule models, and it is at this point that they are employed.

We can view the DESCRIPTION part of the rule model selected earlier as a set of predictions about the likely content of the new rule. In these terms the next step is to see how well each interpretation fulfills those predictions. Note, for example, that the last line of the premise description in Figure 9-6 "predicts" that a rule about category of organism will contain the attribute PORTAL and the third clause of Rule 383 fulfills this prediction. Each interpretation is scored (employing the "strength of advice" number in the rule model) according to how many predictions it fulfills, yielding the *prediction satisfaction score.* This score is then combined with the text score to indicate the most likely interpretation. Because more weight is given to the prediction satisfaction score, the system tends to "hear what it expects to hear."

Rule Interpretation: Sources of Performance

While our approach to natural language is very simple, the overall performance of the interpretation process is adequate. The problem is made easier, of course, by the fact that we are dealing with a small amount of text in a restricted context and written in a semiformal technical language, rather than with large amounts of text in unrestricted dialogue written in unconstrained English. Even so, the problem of interpretation is substan-

tial. TEIRESIAS' performance is based on the application of the ideas noted above (Section 9.6.1), notably the ideas of building expectations and model-based understanding. Its performance is also based on the use of two additional techniques: the intersection of data-driven and model-driven processing, and the use of multiple sources of knowledge.

First, the interpretation process proceeds in what has been called the *recognition mode*: it is the intersection of a bottom-up (data-directed) process (the interpretations suggested by the connotations of the text) with a top-down (goal-directed) process (the expectations set up by the choice of a rule model). Each process contributes to the end result, but it is the combination of them that is effective. This intersection of two processing modes is important when the interpretation techniques are as simple as those employed here, but the idea is more generally applicable as well. Even with more powerful interpretation techniques, neither data-directed nor goal-directed processing is in general capable of eliminating all ambiguity and finding the correct answer. By moving from both directions, top-down and bottom-up, we make use of all available sources of information, resulting in a far more focused search for the answer. This technique is applicable across a range of different interpretation problems, including those of text, vision, and speech.

Second, in either direction of processing, TEIRESIAS uses a number of different sources of knowledge. In the bottom-up direction, for example, distinct information about the appropriate interpretation of the text comes from (a) the connotations of individual words (interpretation of each piece of data), (b) the function template (structure for the whole interpretation), and (c) internal consistency constraints (interactions between data points), as well as several other sources [see Davis (1976) for the full list]. Any one of these knowledge sources alone will not perform very well, but acting in concert they are much more effective [a principle developed extensively in the HEARSAY system (Reddy et al., 1973)].

The notion of program-generated expectations is also an important source of power, since the selection of a particular rule model supplies the focus for the top-down part of the processing. Finally, the idea of model-based understanding offers an effective way of using the information in the rule model to effect the top-down processing.

Thus our relatively simple techniques supply adequate power because of the synergistic effect of multiple, independent sources of knowledge, because of the focusing and guiding effect of intersecting data-directed and goal-directed processing, and because of the effective mechanism for interpretation supplied by the idea of model-based understanding.

9.6.4 Phase III: Modifying the Interpretation

TEIRESIAS has a simple rule editor that allows the expert to modify existing rules or (as in our example) to indicate changes to the system's at-

tempts to understand a new rule.[7] The editor has a number of simple heuristics built into it to make the rule modification process as effective as possible. In dealing with requests to change a particular clause of a new rule, for instance, the system reevaluates the alternative interpretations, taking into account the rejected interpretation (trying to learn from its mistakes) and making the smallest change possible (using the heuristic that the original clause was probably close to correct). In our example, this succeeds in choosing the correct clause next (the second choice shown in Figure 9-7c).

There are also various forms of consistency checking available. One obvious but effective constraint is to ensure that each word of the text is interpreted in only one way. In the trace shown earlier, for instance, accepting the new interpretation of clause 1 means clause 2 must be spurious, since it attempts to use the word infection in a different sense.

9.6.5 Phase IV: Second-Guessing, Another Use of the Rule Models

After the expert indicates that TEIRESIAS has correctly understood what he or she has written, the system checks to see if *it* is satisfied with the content of the rule. The idea is to use the rule model to see how well this new rule "fits into" the system's model of its knowledge; i.e., does it "look like" a typical rule of the sort expected?

In the current implementation, an incomplete match between the new rule and the rule model triggers a response from TEIRESIAS. Recall the last line of the premise description in the rule model of Figure 9-6:

((SITE MEMBF SAME) (INFECTION SAME) (PORTAL SAME) 1.23))

This indicates that when the culture SITE for the patient appears in the premise of a rule of this sort, then INFECTION type and organism PORTAL of entry typically appear as well. Note that the new rule in the example has the first two of these, but is missing the last, and the system points this out.

If the expert agrees to the inclusion of a new clause, TEIRESIAS attempts to create it. Since in this case the agreed-on topic for the clause was the portal of entry of the organism, this must be the attribute to use. The rule model suggests which predicate function to use (SAME, since that is the one paired with PORTAL in the relevant line of the rule model), and the template for this function is retrieved. It is filled out in the usual way, except that TEIRESIAS checks the record of the consultation when seeking items to fill in the template blanks. In this case only a value is still missing. Note that since the expert indicated that the portal of entry was

[7]Much of the editor has subsequently been incorporated into EMYCIN—see Chapter 15.

GI, TEIRESIAS uses this as the value for PORTAL. The result is a plausible guess, since it ensures that the rule will in fact work for the current case (note this further use of the debugging in context idea). It is not necessarily correct, of course, since the desired clause may be more general, but it is at least a plausible attempt.

It should be noted that there is nothing in this concept of second-guessing that is specific to the rule models as they are currently designed, or indeed to associative triples or rules as a knowledge representation. The fundamental point (as mentioned above) is testing to see how the new knowledge "fits into" the system's current model of its knowledge. At this point the system might perform any kind of check, for violations of any established prejudices about what the new chunk of knowledge should look like. Additional kinds of checks of rules might concern the strength of the inference, number of clauses in the premise, etc. In general, this second-guessing process can involve any characteristic that the system may have "noticed" about the particular knowledge representation in use.

Note also that this use of the rule model for second-guessing is quite different from the first use mentioned—guiding the understanding of English. Earlier we were concerned about interpreting text and determining what the expert actually said; here the task is to see what the expert plausibly *should have* said. Since, in assembling the rule models, TEIRESIAS may have noticed regularities in the reasoning about the domain that may not yet have occurred to the expert, the system's suggestions may conceivably be substantive and useful.

Finally, all this is in turn an instance of the more general notion of using meta-level knowledge in the process of knowledge acquisition: TEIRESIAS does not simply accept the new rule and add it to the knowledge base; it instead uses the rule model to evaluate the new knowledge in light of its current knowledge base. In a very simple way, learning is effected as a process of examining the relationships between what is already known and the new information being taught.

9.6.6 Phase V: Final Checkout, Bookkeeping, Rerunning the Consultation

When both the expert and TEIRESIAS are satisfied, there is one final sequence of tests to be performed, reflecting once again the benefit of knowledge acquisition in context. At this point TEIRESIAS examines several things about the rule, attempting to make sure that it will in fact fix the problem uncovered. In this case, for instance, the action of the new rule should be a conclusion about category, the category mentioned should be *Enterobacteriaceae,* and the conclusion should be affirmative. The premise should not contain any clauses that are sure to fail in the context in which the rule will be invoked. All these are potential sources of error that would make it obvious that the rule will not fix the bug.

There are also a number of straightforward bookkeeping tasks to be performed, including hooking the new rule into the knowledge base so that it is retrieved and invoked appropriately (in this case it gets added to the list of rules that conclude about category),[8] and tagging it with information that will make it easier to maintain the large and constantly changing body of rules (e.g., the name of the rule author, date of creation, author's justification for adding the rule, a pointer to the consultation that prompted its creation).

At this point, the system also performs any necessary recomputation of rule models. The operation is very fast, since it is clear from the action part of the rule which models may need to be recomputed, and the EXAMPLES part of the model then supplies the names of the other relevant rules. TEIRESIAS then reruns the performance program as a subprocess, and checks the results to see if all of the problems have been repaired.

9.7 Other Uses for the Rule Models

Two other uses have been developed for the rule models, which demonstrate capabilities made possible by meta-level knowledge.

9.7.1 "Knowing What You Know"

As described in Part Six, MYCIN has the ability to answer simple natural language questions about the knowledge base. In response to a question such as "How do you determine the identity of an organism causing an infection?" MYCIN would originally have printed the relevant rules. But a rule model, as a generalization of an entire class of rules, answers the question, too. Figure 9-8 shows one example of MYCIN's capabilities after rule models had been added. By simply "reading" the rule model to the user, TEIRESIAS can supply an overview of the knowledge in the relevant rules. This suggests the structure of global trends in the knowledge of the expert who assembled the knowledge base, and thus helps to make clear the overall approach of the system to a given topic.

[8]Note that these tests require the ability to dissect and partially evaluate the rule. The same function template that is used as a pattern for constructing rules is also used as a guide in this dissection and partial evaluation process.

```
** HOW DO YOU DECIDE THAT AN ORGANISM IS PSEUDOMONAS
AERUGINOSA?

Rules which conclude that the identity of the organism is
pseudomonas-aeruginosa generally use one or more of the following pieces
of information:
        the site of the culture
        the gram stain of the organism
        the morphology of the organism
Furthermore, the following relationships hold:
        The gram stain of the organism and the morphology of the
        organism tend to appear together in these rules.
RULE184, RULE116, RULE047, RULE085, and RULE040 conclude that the
        identity of the organism is pseudomonas-aeruginosa.
Which of these do you wish to see?
**
```

FIGURE 9-8 "Knowing what you know."

9.7.2 "Knowing What You Don't Know"

Another use of the models demonstrates that, in a primitive fashion, they give TEIRESIAS a model of what it *doesn't* know. There are models in the current system made from between 2 (the defined minimum) and 35 rules. We have defined a metric to measure the *strength* of a model, based on both the total number of rules from which the model was constructed and the strength of the inference of each of those rules. The entire model set is kept ordered from weakest to strongest, giving the system some indication of its likely competence on a range of subjects. In a very primitive way, it thus gains knowledge of where it is ignorant.

This makes possible a new option for the expert. Instead of adding a rule to fix a bug, the expert can ask the system what it would like to learn. (See Figure 9-9.) The system cycles through the rule models in order, indicating the weakest topics first. This is, of course, a first-order solution to the problem of giving the system an indication of its areas of ignorance. A better solution would supply an indication of how much the system knows about a subject, compared with how much there is to know. There surely are subjects for which three or four rules exhaust the available knowledge, while for others a hundred or more rules may not suffice. The issue is related to work described by Carbonell and Collins (1973) on closed versus open sets. That paper offers some interesting strategies for allowing a program to decide when it is ignorant and how it might reason in the face of the inability to store every fact about a given topic.

There appear to be no easy ways to deduce the incompleteness of the knowledge base using only the information stored in it. It is not valid to say, for instance, that there ought to be even a single rule for every attribute (how could a patient's name be deduced?). Nor is there a well-defined set of attributes for which no rules are likely to exist. Nor is it clear what sort

```
--** WHAT WOULD YOU LIKE TO KNOW?
Answer
  -Y for yes
  -N for no, or
  -Q to quit this.
There are only two rules for concluding that the identity of an
organism might be enterococcus. Would you like to give me a
rule for that?
++** N
There are only 2 rules for concluding that an infection might
be infective-endocarditis. Would you like to offer a rule for
that?
++** Y
The new rule will be called RULE384
  If (1)
```

FIGURE 9-9 "Knowing what you don't know."

of information would allow the incompleteness to be deduced. The issue is a significant one, since a good solution to the problem would not only give TEIRESIAS a better grasp of where the performance program was weak but also provide several important capabilities to the performance program itself. It would, for example, permit the use of the "if it were true I would know" heuristic of Carbonell and Collins (1973). Roughly restated, this says "If I know a great deal about subject *S*, and fact *F* concerns an important aspect of *S*, then if I don't already know that *F* is true, it's probably false." Thus in certain circumstances a lack of knowledge about the truth of a statement can plausibly be used as evidence suggesting that the statement is false. This is another useful form of meta-level knowledge.

9.8 Assumptions and Limitations

The work reported here can be evaluated with respect to both the utility of its approach to knowledge acquisition and its success in implementing that approach.

9.8.1 The Approach

As noted, our approach involves knowledge transfer that is interactive, that is set in the context of a shortcoming in the knowledge base, and that transfers a single rule at a time. Each of these has implications about TEIRESIAS's range of applicability.

Interactive knowledge transfer seems best suited to task domains in-

volving problem solving that is entirely or primarily a high-level cognitive task, with a number of distinct, specifiable principles. Consultations in medicine or financial investments seem to be appropriate domains, but the approach would not seem well suited to those parts of, say, speech understanding or scene recognition in which low-level signal processing plays a significant role.

The transfer of expertise approach presents a useful technique for task domains that do not permit the use of programs (like those noted in Section 9.2) that autonomously induce new knowledge from test data. The autonomous mode may most commonly be inapplicable because the data for a domain simply don't exist yet. In quantitative domains [such as mass spectrum analysis (Buchanan and Feigenbaum, 1978)] or synthesized ("toy") domains [such as the line drawings in Hayes-Roth and McDermott (1977)], a large body of data points is easily assembled. This is not currently true for many domains; consequently induction techniques cannot be used. In such cases interactive transfer of expertise offers a useful alternative.[9]

Knowledge acquisition in context appears to offer useful guidance wherever knowledge of the domain is as yet ill-specified. The context of the interaction need not be a shortcoming in the knowledge base uncovered during a consultation, however, as it was here. Our recent experience suggests that an effective context is also provided by examining certain subsets of rules in the knowledge base and using them as a framework for specifying additional rules. The overall concept is limited, however, to systems that already have at least some minimal amount of information in their knowledge bases. Prior to this, there may be insufficient information to provide any context for the acquisition process.

Finally, the rule-at-a-time approach is a limiting factor. The example given earlier works well, of course, because the bug was manufactured by removing a single rule. In general, acquiring a single rule at a time seems well suited to the later stages of knowledge base construction, in which bugs may indeed be caused by the absence of one or a few rules. We need not be as lucky as in the example, in which one rule repaired three bugs; the approach will also work if three independent bugs arise in a consultation. But early in knowledge base construction, when large subareas of a domain are not yet specified, it appears more useful to deal with groups of rules or, more generally, with larger segments of the basic task [as in Waterman (1978)].

In general then, the interactive transfer of expertise approach seems well suited to the later stages of knowledge base construction for systems performing high-level tasks, and offers a useful technique for domains where extensive sets of data points are not available.

---

[9]Where the autonomous induction technique can be used, it offers the interesting advantage that the knowledge we expect the system to acquire need not be specified ahead of time, indeed not even known. Induction programs are in theory capable of inducing new information (i.e., information unknown to their author) from their set of examples. Clearly, the interactive transfer of expertise approach requires that the expert know and be able to specify precisely what it is the program is to learn.

9.8.2 The Program

Several difficult problems remained unsolved in the final implementation of the program. There is, for instance, the weakness of the technique of natural language understanding. There is also an issue with the technique used to generate the rule models. Model generation could be made more effective even without using a different approach to concept formation. Although an early design criterion suggested keeping the models transparent to the expert, making the process interactive would allow the expert to evaluate new patterns as they were discovered by TEIRESIAS. This might make it possible to distinguish accidental correlations from valid interrelations and might increase the utility and sophistication of TEIRESIAS's second-guessing ability. Alternatively, more sophisticated concept formation techniques might be borrowed from existing work.

There is also a potential problem in the way the models are used. Their effectiveness both in guiding the parsing of the new rule and in second-guessing its content is dependent on the assumption that the present knowledge base is both correct and a good basis for predicting the content of future rules. Either of these can at times be false, and the system may then tend to continue stubbornly down the wrong path.

There is also the difficult problem of determining the impact of any new or changed rule on the rest of the knowledge base, as discussed in Chapter 8, which we have considered only briefly. One difficulty (avoided in the work described in Chapter 8) involves establishing a formal definition of inconsistency for inexact logics, such as CF's (see Chapter 11), since, except for obvious cases (e.g., two identical rules with different strengths), it is not clear what constitutes an inconsistency. Once the definition is established, we would also require routines capable of uncovering them in a large knowledge base. This can be attacked by using an incremental approach (i.e., by checking every rule as it is added, the knowledge base is kept consistent and each consistency check is a smaller task), but the problem is substantial.

9.9 Conclusions

Each of the ideas reviewed above offers some contribution toward achieving the two goals set out at the beginning of this chapter: the development of a methodology of knowledge base construction via transfer of expertise, and the creation of an intelligent assistant to aid in knowledge acquisition. These ideas provide a set of tools and ideas to aid in the construction of knowledge-based programs and represent some new empirical techniques of knowledge engineering. Their contribution here may arise from their

potential utility as case studies in the development of a methodology for this discipline.

Knowledge acquisition in the context of a shortcoming in the knowledge base, for instance, has proved to be a useful technique for achieving transfer of expertise, offering advantages to both the expert and TEIRESIAS. It offers the expert a framework for the explication of a new chunk of domain knowledge. By providing a specific example of the performance program's operation and forcing the expert to be specific in his or her criticism, it encourages the formalization of previously implicit knowledge. It also enables TEIRESIAS to form a number of expectations about the knowledge it is going to acquire and makes possible several checks on the content of that knowledge to ensure that it would in fact fix the bug. In addition, because the system has a *model of its own knowledge,* it is able to determine whether a newly added piece of knowledge "fits into" its existing knowledge base.

A second contribution of the ideas reviewed above lies in their ability to support a number of intelligent actions on the part of the assistant. While those actions have been demonstrated for a single task and system, it should be clear that none of the underlying ideas are limited to this particular task or to associative triples or rules as a knowledge representation. The foundation for many of these ideas is the concept of meta-level knowledge, which has made possible a program with a limited form of introspection.

The idea of *model-based understanding,* for instance, found a novel application in the fact that TEIRESIAS has a model of the knowledge base and uses this to guide acquisition by interpreting the model as predictions about the information it expects to receive.

The idea of *biasing the set of models* to be considered offers a specific mechanism for the general notion of *program-generated expectations* and makes possible an assistant whose understanding of the dialogue is more effective.

TEIRESIAS is able to second-guess the expert with respect to the content of the new knowledge by using its models to *see how well the new piece of knowledge "fits into" what it already knows.* An incomplete match between the new knowledge and the system's model of its knowledge prompts it to make a suggestion to the expert. With this approach, learning becomes more than simply adding the new information to the knowledge base; TEIRESIAS examines as well the relationship between the new and existing knowledge.

The concept of meta-level knowledge makes possible *multiple uses of the knowledge in the system:* information in the knowledge base is not only used directly (during the consultation) but also examined and abstracted to form the rule models.

TEIRESIAS also represents a synthesis of the ideas of model-based understanding and learning by experience. Although both of these have been developed independently in previous AI research, their combination

produces a novel sort of feedback loop (Figure 9-10). Rule acquisition relies on the set of rule models to effect the model-based understanding process. This results in the addition of a new rule to the knowledge base, which in turn prompts the recomputation of the relevant rule model(s).[10]

This loop has a number of interesting implications. First, performance on the acquisition of the next rule may be better because the system's "picture" of its knowledge base has improved—the rule models are now computed from a larger set of instances, and their generalizations are more likely to be valid. Second, since the relevant rule models are recomputed each time a change is made to the knowledge base, the picture they supply is kept constantly up to date, and they will at all times be an accurate reflection of the shifting patterns in the knowledge base. This is true as well for the trees into which the rule models are organized: they too grow (and shrink) to reflect the changes in the knowledge base.

Finally, and perhaps most interesting, the models are not handcrafted by the system architect or specified by the expert. They are instead formed by the system itself, and formed as a result of its experience in acquiring rules from the expert. Thus, despite its reliance on a set of models as a basis for understanding, TEIRESIAS's abilities are not restricted by the existing set of models. As its store of knowledge grows, old models can become more accurate, new models will be formed, and the system's stock of knowledge about its knowledge will continue to expand. This appears to be a novel capability for a model-based system.

[10]The models are recomputed when any change is made to the knowledge base, including rule deletion or modification, as well as addition.

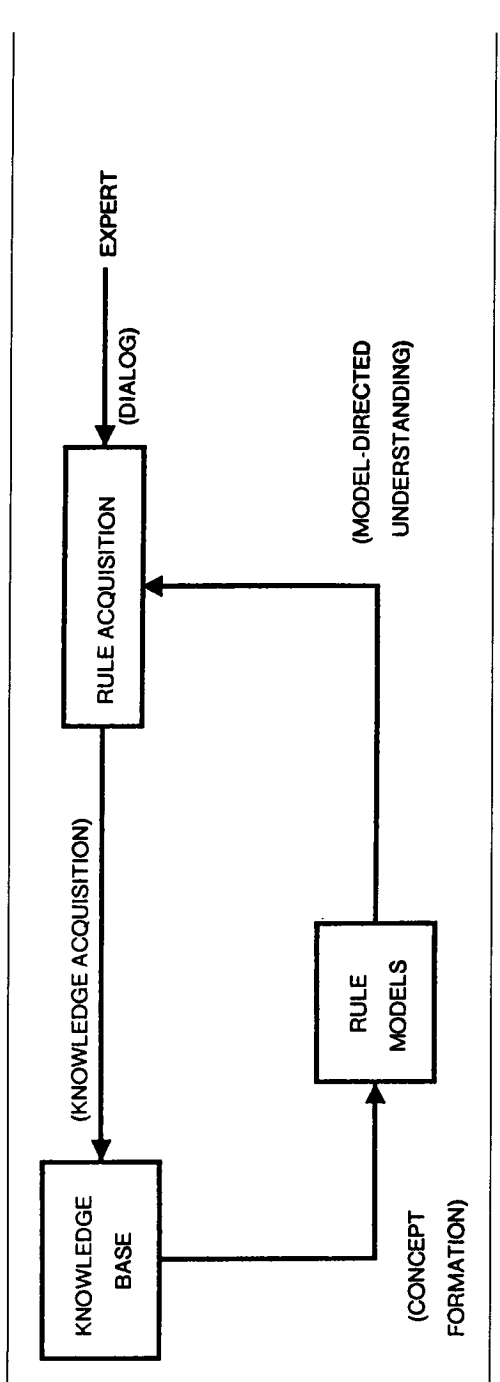


FIGURE 9-10 Model-directed understanding and learning by experience combine to produce a useful feedback loop.