Interactive Transfer of Expertise: Acquisition of New Inference Rules

Randall Davis*

Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A.

ABSTRACT

TEIRESIAS *is a program designed to provide assistance on the task of building knowledge-based systems. It facilitates the interactive transfer of knowledge from a human expert to the system, in a high level dialog conducted in a restricted subset of natural language. This paper explores an example of* TEIRESIAS *in operation and demonstrates how it guides the acquisition of new inference rules. The concept of* meta-level knowledge *is described and illustrations given of its utility in knowledge acquisition and its contribution to the more general issues of creating an intelligent program.*

1. Introduction

Where much early work in artificial intelligence was devoted to the search for a single, powerful, domain-independent problem solving methodology (e.g., GPS [14]), more recent 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 [11], MACSYMA [13]) is traditionally assembled by hand, an ongoing task that typically involves numerous man-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 task. The result is a system called TEIRESIAS[1] [5–8], a large INTERLISP [19] program designed to offer assistance in the interactive transfer of knowledge from a human expert to the knowledge base of a high performance program (Fig. 1).

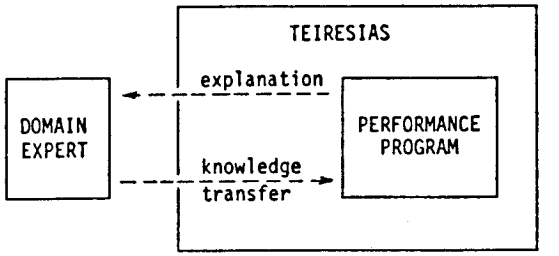


FIG. 1. Interaction between the expert and the performance program is facilitated by TEIRESIAS.

Information flow from right to left is labelled *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 program already knows and how it used that knowledge. Information flow from left to right is labelled *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 and empirical methods for knowledge base construction and maintenance, and sought 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 paper describes a number of the key ideas in the development of TEIRESIAS and discusses their implementation in the context a specific task (acquisition of new inference rules[2]) for a specific performance program (a rule-based computer consultant). While the discussion deals with one particular task, system and knowledge representation, several of the main ideas are applicable to more general issues concerning the creation of intelligent programs.

2. Meta-Level Knowledge

A central theme that runs through this and related papers [5–8] is the concept of *meta-level knowledge*. This takes several different forms as its use is explored, but can be summed up generally by saying that a program can "know what it knows". That is, a program can not only use its knowledge directly, but may also be able to examine it, abstract it, reason about it, and direct its application.

* Author's current address: 545 Technology Square, MIT, Cambridge, MA 02138, U.S.A.

This work was supported in part by the Advanced Research Projects Agency under ARPA Order 2494; by a Chaim Weizmann Postdoctoral Fellowship for Scientific Research, and by grant MCS 77-02712 from the National Science Foundation. It was carried out on the SUMEX Computer System, supported by the NIH Grant RR-00785.

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

[2] Acquisition of new conceptual primitives from which rules are built is discussed in [7], while the design and implementation of the explanation capability suggested in Fig. 1 is discussed in [5].

To see in general terms how this might be accomplished, recall that one of the principal problems of AI is the question of representation and use 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, and using some of these same techniques to describe the program itself.

The resulting system contains both *object-level* representations describing the external world, and *meta-level* representations which describe the internal world of representations. As the discussion of "rule models" in Section 7 will make clear, such a system has a number of interesting capabilities.

3. 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 a program like DDT, which 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 [16] and INTERLISP's break package [19] are a step up from this since they function at the level of programming languages like SAIL and INTERLISP.

The explanation capabilities in TEIRESIAS, in particular the "how" and "why" commands (see [5] and [9] for examples), represent another step, since they function at the level of the control structure of the application program. The guided debugging which TEIRESIAS can also provide (illustrated in Section 6) 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. It does this by requesting from the expert an opinion on the validity of a few selected rules from among the many that were invoked.

Finally, at the most sophisticated level are knowledge-rich debuggers like the one found in [2]. Here the program is active, high-level, and informed about the application domain, and is 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 acquisition 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, e.g., [3] and [12]). 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 [21] relied somewhat more heavily on user interaction. There the teacher was responsible for providing an appropriate sequence of examples (and non-examples) 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 knowledge on its own, but instead tries to "listen" as attentively as possible and comment appropriately, to help the expert augment the knowledge base. It thus requires the strongest degree of 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 basic *problem-solving paradigm* and *degree of 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 assumed 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 (at the conceptual level) that he can (a) understand the system's behavior in terms of them and (b) use them to codify his own knowledge. This insures that the expert understands system performance well enough to know what to correct and can then express the required knowledge, i.e., he can "think" in those terms. Thus part of the task of establishing the link shown in Fig. 1 involves insulating the expert from the details of implementation, by establishing a discourse at a level high enough that we do not end up effectively having to teach him how to program.

4. Design of the Performance Program

4.1. Program architecture

Fig. 2 shows a slightly more detailed picture of the sort of performance program that TEIRESIAS is designed to help construct. (The performance program described here is modelled after the MYCIN program [17, 9], which provided the context

within which TEIRESIAS was actually developed. We have abstracted out here just the essential elements of MYCIN's design.) 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.

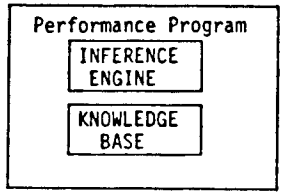


FIG. 2. Architecture of the performance program.

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 would be to augment the knowledge base of a program whose control structure (inference engine) was assumed 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 and be assured that the discussion will have to deal only with issues of domain specific expertise (rather than with questions of programming and control structures). Second, if all of the task-specific knowledge has been kept in the knowledge base, then it should be possible to remove the current knowledge base, "plug in" another, and obtain a performance program for a new task.[3] 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 which motivates the important assumption noted in Section 3 that the control structure and knowledge representation are comprehensible to the expert, at least at the conceptual level.

In this discussion we assume the knowledge base contains information about selecting an investment in the stock market; the performance program thus functions as an investment consultant. MYCIN, of course, deals with infectious disease diagnosis and therapy selection, and the rules and dialog shown later dealt with that subject initially. The topic has been changed to keep the discussion phrased in terms familiar to a wide range of readers, and to emphasize that neither the problems attacked nor the solutions suggested are restricted to a single domain of application or performance program design. The dialog is a real example of TEIRESIAS in action with a few words substituted in a medical example: e.g., *E. coli* became *AT&T*, *infection* became *investment*, etc.

An example of the program in action is shown in Section 6. The program interviews the user, requesting various pieces of information that are relevant to selecting the most appropriate investment, then prints its recommendations. In the remainder of this paper the "user" will be an expert running the program in order to challenge it, offering it a difficult case, and observing and correcting its performance.

4.2. The knowledge base

The knowledge base of the performance program contains a collection of decision rules of the sort shown in Fig. 3. (The rule is stored internally in the INTERLISP form, the English version is generated from that with a simple template-directed mechanisms.) Each rule is a single "chunk" of domain specific information indicating an *action* (in this case a conclusion) which is justified if the conditions specified in the *premise* are fulfilled.

The rules are judgmental, i.e., they make inexact inferences. In the case of the rule in Fig. 3, for instance, the evidence cited in the premise is enough to assert the conclusion shown with only a weak degree of confidence (0.4 out of 1.0). These numbers are referred to as *certainty factors*, and embody a model of confirmation described in detail in [18]. The details of that model need not concern us here; we need only note that a rule typically embodies an inexact inference rather than an exact rule.

```
RULE 027

If   [1.1] the time scale of the investment is long-term,
     [1.2] the desired return on the investment is greater than 10%, and
     [1.3] the area of the investment is not known,
then AT&T is a likely (0.4) choice for the investment.

PREMISE  ($AND (SAME OBJCT TIMESCALE LONG-TERM)
               (GREATER OBJCT RETURNRATE 10)
               (NOTKNOWN OBJCT INVESTMENT-AREA))

ACTION   (CONCLUDE OBJCT STOCK-NAME AT&T 0.4)
```

FIG. 3. Example of a rule.

Finally, a few points of terminology. The premise is a Boolean combination of one or more *clauses*, each of which is constructed from a *predicate function* with an *associative triple* (*attribute*, *object*, *value*) as its argument. For the first clause in Fig. 3, for example, the predicate function is SAME, and the triple is "*timescale* of *investment* is *long-term*". (The identifier OBJCT is used as a placeholder for the

[3] Two experiments of this sort have been performed with the MYCIN system, and suggest that this sort of "plug compatibility" of knowledge bases is a realistic possibility for a range of tasks.

specific object to be referred to; the actual binding is established each time the rule is invoked.)

4.3. The inference engine

The rules are invoked in a simple backward-chaining fashion that produces an exhaustive depth-first search of an and/or goal tree (Fig. 4). Assume that the program is attempting to determine which stock would make a good investment. It retrieves (the precomputed list of) all rules which make a conclusion about that topic (i.e., they mention STOCK-NAME in their action), and invokes each one in turn, evaluating each premise to see if the conditions specified have been met. For the example shown in Fig. 4, this means first determining what the timescale of the investment ought to be. This is in turn set up as a subgoal, and the process recurs.

The search is thus depth-first (because each premise condition is thoroughly explored in turn); the tree that is sprouted is an and/or goal tree (because rules may have OR conditions in their premise); and the search is exhaustive (because the rules are inexact, so that even if one succeeds, it was deemed to be a wisely conservative strategy to continue to collect all evidence about the subgoal.)

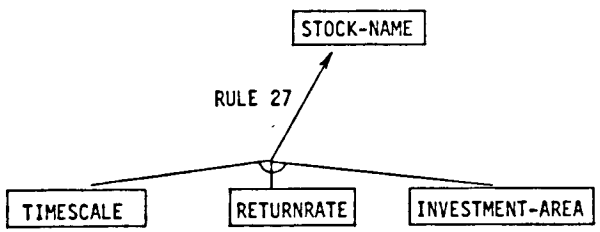


FIG. 4. Control structure of the performance program is an exhaustive, depth-first search of an and/or goal tree.

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 to enable him to express new knowledge for the performance program. That level of understanding is well supplied by the information in Sections 4.2 and 4.3, together with an equally brief summary of the model of confirmation described in [18]. This should make plausible the assumption in Section 3 that the expert might indeed be able to understand these things sufficiently without having to deal with details of implementation. Note too that TEIRESIAS's basic design and the notion of interactive transfer of expertise does not depend on this particular control structure, only on the (non-trivial) assumption that an equally comprehensible explanation can be found for whatever control structure is actually used in the inference engine.

5. A Word about Natural Language

Natural language has not been a major focus of this work, and for the most part the simplest techniques have been used 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 the occasional appearance of phrases like "a area"). Where answers are free text (rather than multiple choice), the system relies primarily on the keyword- and template-oriented techniques described in Section 7.3.4. There is no parser in the system, in part to minimize processing time, and in part because users often give ungrammatical responses.

6. Example of TEIRESIAS in Operation

In the interaction shown below, the expert is challenging the investment advisory program 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 to make possible an acquisition example that displays 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 below, computer output is in mixed upper and lower case in this font, user responses follow a double asterisk and are in **BOLD CAPITALS**, and commentary appears as {*indented italics inside braces*}. The example is also divided into five phases for later reference.

Investment Advisory Program

User?
**** RD**

Hello, Randy.

{*The expert identifies himself to the program, and then, responding to the program's inquiries, describes a particular case as a test of performance.*}

(1) Investor's name?
****FRED SMITH**

(2) Age?
****40**

(3) Present taxable income?
****40,000**

(4) Number of dependents?
**3

(5) Occupation?
**LAWYER

(6) Amount of previous investment experience? (slight = less than one year; moderate = 2 to 4 years; extensive = more than 4 years)
**MODERATE

(7) Does the investor subscribe to the Wall Street Journal, or any securities market newsletters?
**Y

.
.
.

{*Most of the consultation with the performance program has been edited out for brevity (see* [5] *for a complete example). The interaction continues with the program printing out its conclusions.*}

The following investments appear to be the most appropriate at this time:
Varian-Corporation
American-Telephone-and-Telegraph

{*At this point* TEIRESIAS *steps in, and since the interaction is with an expert (rather than an ordinary user), it requests an evaluation of the performance program's results.*

From here on the dialog takes place between TEIRESIAS *and the expert, with the performance program in the background being manipulated by* TEIRESIAS.}

{Phase I: tracking down the bug}

{*The expert indicates that the results are incorrect . . .*}

Are all these correct?
++**N

{. . . *and then specifies the problems.*}

Are there any missing?
++**Y
Which one(s)?
++** DIGITAL-EQUIPMENT DATA-GENERAL
Are there any that shouldn't have appeared?
++** Y
Which one(s)?
++** AT&T

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

{*The first option allows the expert to use the explanation facilities described in* [5] *to track down the problem. The facility indicated by the second option is based on a model of the control structure of the performance program that allows a simple algorithmic search for the error, and will be demonstrated here. The third is for human engineering purposes, and allows the expert to sidestep this whole process if desired.*}

We'll start with the AT&T first.

{*Heuristics derived from our experience in debugging rule-based systems are built into* TEIRESIAS. *One such heuristic suggests that the best approach is to* (1) *track down one problem at a time, and* (2) *start with a result that shouldn't have appeared (if there is one). Experience indicates that this class of bug is often a manifestation of a deeper bug causing several problems, so curing it will often solve several problems at once.*}

The following rules were used in deducing that the investment should be AT&T:

RULE 027

Since [1.1] the area of the investment is not known, and
[1.2] the desired return on the investment is greater than 10%, and
[1.3] the time scale of the investment is long-term,
then AT&T is a likely (0.4) choice for the investment.

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

{*The expert indicates that this rule should have failed, so the debugging continues by looking further back in the chain. Had he indicated that rule* 27 *had been invoked properly, then* TEIRESIAS *would have indicated that the problem was probably a missing rule which tended to* ***rule out*** *AT&T.*}

No conclusions were reached concerning the area of the investment.

Should it have been possible to conclude an area?
++** Y

{*The error might have been a failure to rule out an investment area.*}

Which one?
++** HIGH TECHNOLOGY

There were no rules successfully invoked in the attempt to deduce that the area of the investment should be high-technology.

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

RULE 116

If (1) the desired return on the investment is greater than 10%,
(2) the time scale of the investment is long-term,
(3) the number of dependents of the client is less than 3,
(4) the age (in years) of the client is less than or equal to 35
then there is evidence (0.4) that the area of the investment should be high-technology.

RULE 050

If (1) the time scale of the investment is long-term,
(2) the desired return on the investment is greater than 10%,
(3) the risk class of the investment is speculative
then there is evidence (0.6) that the area of the investment should be high-technology.

Are all these rules correct?
++** Y
Should any of them have been successfully invoked?
++** N

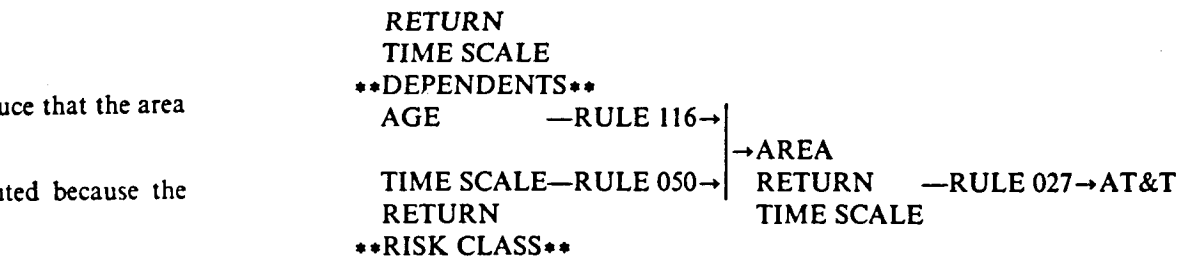
{This would have been the case if one of the starred premise clauses were false because it was "stepped on" by yet another rule further back in the chain.}

Is there a rule missing?
++** Y

Ok, then I need a rule that would allow me to deduce that the area of the investment should be high-technology.

Would you like to give me such a rule now?
++** Y
Good . . .

{To review briefly: the expert is tracking down the reason why the system incorrectly concluded that the stock to invest in was AT&T. Shown below is the sequence of rules which were involved:



AT&T appeared because the system executed Rule 27 successfully.
This occurred in part because investment area was unknown.
Investment area was unknown because all rules to conclude it (Rules 50 and 116) failed.
Since the expert believes all existing rules are correct, the knowledge base must be missing a rule that will conclude about investment area, in particular, that the area should be high-technology. Such a rule will cause area to be known, inhibiting Rule 27, fixing at least this first problem.
Note how sharply focussed the issue has become, for both the expert and TEIRESIAS, *as a result of the large amount of contextual information available. The expert is not being asked to "discuss the stock market"; instead the system can at this point say*
. . . I need a rule that would allow me to deduce that the area of the investment should be high technology.
and it must be a rule which is invokable in the current situation. Thus it is easier for the expert to specify a rule that may never before 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. The information provides a number of expectations about the content of the rule which are used 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 RULE 383

If: 1. **THE CLIENT'S INCOME TAX BRACKET IS 50%**
and 2. **THE CLIENT IS FOLLOWING UP ON MARKET TRENDS CAREFULLY**
and 3.
Then: 1. **THERE IS EVIDENCE (0.8) THAT THE INVESTMENT AREA SHOULD BE HIGH TECHNOLOGY**
and 2.

This may take a few moments . . .

{*Generally on the order of* 10 *to* 30 *seconds of cpu time.*}

This is my understanding of your rule:

RULE 383

If (1) The client's income-tax bracket is 50%, and
(2) The market has followed an upward trend recently, and
(3) The client manages his assets carefully,
then there is evidence (0.8) that the area of the investment should be high-technology

{TEIRESIAS*'s initial interpretation is not entirely correct (and the example was chosen in part for this reason). The first line of text and the conclusion have been interpreted correctly, but the second line has been turned into two clauses (numbers* 2 *and* 3*) in the system's initial attempt at understanding. (The expert occassionally types what should be two separate clauses as one line of text. The system is capable of handling this, but this same capability at times produces spurious clauses like No.* 3*).*

This level of performance results primarily from the system's simple word-by-word approach to meaning. While the result is in part incorrect, it will become clear below that the interpretations are both "reasonable" and "consistent" in light of the techniques used.}

{**Phase III: modifying the interpretation**}

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

{*The expert indicates there is a problem with the interpretation, and uses* TEIRESIAS*'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 2**

Suppose (2) were changed to read:
(2) The client follows the market carefully
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:
(3) The client manages his assets carefully
Correct?
++** **Y**

{TEIRESIAS *enforces certain consistency constraints on its interpretation of the English text. One of these constraints indicates that if the new interpretation of clause No.* 2 *is correct, then clause No.* 3 *must have arisen spuriously (since, among other things, it requires a different interpretation for the word* carefully).}

Anything else?
++**?
(1) the client's income-tax bracket is 50%
and (2) the client follows the market carefully
and (3) ⟨deleted⟩

Anything else?
++** **N**

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

This is my understanding of your rule:

RULE 383

If: (1) the client's income-tax bracket is 50%, and
(2) the client follows the market carefully,
then there is evidence (0.8) that the area of the investment should be high-technology.

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

{*This is the standard test of comprehension: the original English text has been turned into a rule coded in* LISP*, and is then retranslated back into English for approval. Compare the results with the expert's original input. Now that the expert is satisfied that* TEIRESIAS *understands the rule that he entered,* TEIRESIAS *checks several things about the rule to see if* **it** *is satisfied with the rule content.*}

{**Phase IV: "second guessing"**}

I hate to criticize, Randy, but did you know that most rules about what the area of an investment might be, that mention:
the income-tax bracket of the client, and
how closely the client follows the market
LSO mention:

[A] The amount of investment experience of the client
Shall I try to write a clause to account for [A]?
++** Y

{*The mechanism that makes possible this sort of "second guessing" by the system is explained in detail below. Briefly, the system maintains a model of its own knowledge. In checking to see how this new piece of information "fits in" to 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 amount of investment experience of the client is moderate
Ok?
++** Y

{*The expert agreed to the inclusion of the new clause and* TEIRESIAS *is able to construct a plausible guess as to its content. Note that it must be possible to invoke the new rule in the current situation, in order to fix the known bug.* TEIRESIAS *uses this fact to guide its construction of the new clause. Details of this process are explained below.*}

Also, most rules about what the area of an investment might be ALSO conclude about:
[A] The desired risk classification of the investment
Shall I try to write a clause to account for [A]?
++** 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.*}

{**Phase V: final checkout, bookkeeping, rerunning the consultation**}

The rule now reads:

RULE 383

If (1) the income-tax bracket of the client is 50%, and
(2) the client follows the market carefully, and
(3) the amount of investment experience of the client is moderate,
then there is evidence (0.8) that the area of the investment should be high-technology.

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

++**THE SYSTEM FAILED TO REALIZE IT COULD CONCLUDE
++**INVESTMENT AREA, AND THIS ALLOWED RULE 027 TO BE
++**INVOKED INAPPROPRIATELY
++**

{*Experience suggests that in building any large knowledge base it is very useful to keep fairly detailed records about the source of various contributions. The new rule is automatically tagged with a range of such information (including author, date, and a record of the consultation which prompted its creation), along with any comments which the expert may care to make for his own future reference.*}

RULE 383 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 client.

.
.
.

{*The consultation is then rerun, but all the previous answers are saved, so the expert can in most cases just watch it proceed. This has been edited out here for brevity.*}

The following investments appear to be the most appropriate at this time:

Varian-Corporation
Data-General
Digital-Equipment

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.*}

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

{*As always, however, the expert gets the last word, and* TEIRESIAS *waits for approval before proceeding.*}

Good, then the problem's been fixed.

{*How is it that three errors disappeared with the addition of one rule? This is explained schematically below:*

RULE 116
RULE 050
RULE 383→AREA—*RULE 027*→AT&T
— RULE 084 →DATA-GENERAL
DIGITAL-EQUIPMENT

*As before, Rules 116 and 50 failed (hence the **'s), but now the new rule, 383, has succeeded, supplying a value for investment area. This in turn means that Rule 27 now fails, as hoped.*

However, Rule 84, which requires a value of area, failed last time because the system was unable to deduce one. Now that Rule 383 supplies one, Rule 84 can fire, and it turns out to yield the missing answers.}

7. How it all Works

7.1. Overview of the main ideas

Before reviewing the trace in more detail, we describe the ideas which make possible the capabilities displayed. The list below serves primarily to name and briefly sketch each in turn; the details are supplied in reviewing the example.

7.1.1. *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 which 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 its utility, consider the difference between asking the expert

> *What should I know about the stock market?*

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 focussed the second question is, and how much easier it to answer.

7.1.2. *Building expectations*

The focussing 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.

7.1.3. *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 selected. This sets the framework in which further interpretation takes place, as that model can then be used as a guide to further processing.

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.

7.1.4. *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 which has a picture of its own knowledge. That is, it not only knows something about a particular domain, but in a primitive sense it knows what it knows, and employs that model of its knowledge in several ways.

7.1.5. *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 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.

7.1.6. *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 type of models it has. As will become clear, the models TEIRESIAS employs are not hand-crafted and static, but are instead formed and continually revised as a by-product of its experience in interacting with the expert.

7.2. Phase I: tracking down the bug

To provide the debugging facility shown, 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 (see [5] for details). This presents the expert with a comprehensible task because (a) the backward-chaining technique used by the performance program is sufficiently straightforward and intuitive, even to a non-programmer;

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-by-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 his responses. Note that at each point in tracking down the error the expert must either approve of the rules invoked and conclusions made, or indicate which one was in error and supply the correction. This is extremely useful in domains where knowledge has not yet been formalized, and the traditional reductionist approach of dissecting reasoning down to observational primitives is not yet well established.[4]

TEIRESIAS further encourages precise comments by keeping the debugging process sharply focussed. For instance, when it became clear that there was a problem with the inability to deduce investment area, the system first asks which area it should have been. It then displays only those rules appropriate to that answer, rather than all of the rules on that topic which were tried.

Finally, consider the extensive amount of contextual information that is now available. The expert has been presented with a detailed example of the performance program in action, he 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 to specify the particular chunk of knowledge which may be missing. This contextual information will prove very useful for TEIRESIAS as well. It is clear, for instance, what the *effect* of invocation of the new rule must be (as TEIRESIAS indicates, it must be a rule that will "deduce that the area of the investment should be high-technology"), and it is also clear what the *circumstances* of its invocation must be (the rule must be invokable for the case under consideration, or it won't repair the bug). Both of these will be seen to be quite useful (see Sections 7.3.3 and 7.6).

7.3. 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 re-translating 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.

7.3.1. *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, then 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 particularly graphic example of this idea can be found in the work on computer vision by Falk [10] in 1970. The task there was the standard one of understanding blocks-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 his work of interest here is the use of a set of *prototypes* for the blocks, prototypes that resembled wire frame models. While it oversimplifies slightly, part of the operation of his 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.

While it was not a part of Falk's system, we can imagine one further refinement in the interpretation process 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.

[4] The debugging process does allow the expert to indicate that while the performance program's results are incorrect, he 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 3).

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).

7.3.2. *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 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, one that could (a) capture the structure, organization, or content of the expert's reasoning, (b) be used to guide the interpretation process, and (c) be used to express expectations about the likely content of the new rule.

Where might we get such a thing? There are interesting regularities 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 are used to characterize a "typical" member of the subset.

Rule models are composed of four parts (Fig. 5). They contain, first, a list of EXAMPLES, the subset of rules from which this model was constructed.

EXAMPLES	the subset of rules which this model describes
DESCRIPTION	characterization of a "typical" member of this subset characterization of the premise characterization of the action which attributes "typically" appear correlations of attributes
MORE GENERAL	pointers to models describing more general
MORE SPECIFIC	and more specific subsets of rules

FIG. 5. Rule model structure.

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 (action) of a rule in this subset, and (b) correlations of attributes appearing in the premise (action).[5]

Note that the central idea is the concept of *characterizing a typical member of the subset*. Naturally, that characterization would 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 Fig. 6. At the root of each tree is the model made from all the rules which conclude about ⟨attribute⟩ (e.g., the INVESTMENT-AREA model), below this are two models dealing with all affirmative and all negative rules (e.g., the INVESTMENT-AREA-IS model), and below this are models dealing with rules which affirm or deny specific values of the attribute.

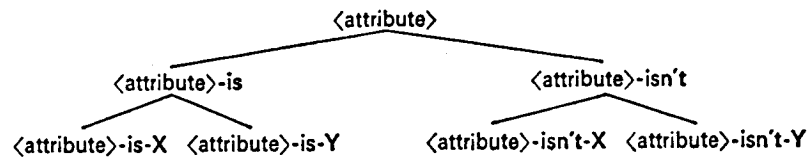


FIG. 6. Organization of the rule models.

These models are not hand-tooled 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 very simple (i.e., statistical) form of concept formation. The combination of TEIRESIAS and the performance program thus presents a system which has a model of its own knowledge, one which it forms itself.

The rule models are the primary example of meta-level knowledge in this paper (for discussion of other forms, see [5] and [8]). This form of knowledge and its generation by the system itself have several interesting implications illustrated in later sections.

Fig. 7 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. See [5] for a full explanation.) 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 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 about what the area of an investment should be mention

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

the attribute *rate of return* in their premise; 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 3.8 (see [5] for precise definition of the quoted terms).

The fourth item in the premise description indicates that when the attribute *rate of return* appears in the premise of a rule in this subset, the attribute *timescale of the investment* "typically" appears as well. As before the predicate functions are those typically associated with the attributes, and the number is an indication of reliability.

INVESTMENT-AREA-IS

```
EXAMPLES      (RULE116 0.3)
              (RULE050 0.7)
              (RULE037 0.8)
              (RULE095 0.9)
              (RULE152 1.0)
              (RULE140 1.0))
DESCRIPTION
  PREMISE     ((RETURNRATE SAME NOTSAME 3.8)
               (TIMESCALE SAME NOTSAME 3.8)
               (TREND SAME 2.8)

               ((RETURNRATE SAME) (TIMESCALE SAME) 3.8)
               ((TIMESCALE SAME) (RETURNRATE SAME) 3.8)
               ((BRACKET SAME) (FOLLOWS NOTSAME SAME)
                                          (EXPERIENCE SAME) 1.5))
  ACTION      ((INVESTMENT-AREA CONCLUDE 4.7)
               (RISK CONCLUDE 4.0)

               ((INVESTMENT-AREA CONCLUDE) (RISK CONCLUDE) 4.7))
MORE-GENL     (INVESTMENT-AREA)
MORE-SPEC     (INVESTMENT-AREA-IS-UTILITIES)
```

FIG. 7. Example of a rule model.

7.3.3. *Choosing a model*

It was noted earlier that tracking down the bug in the knowledge base provides useful context, and, among other things, serve 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 which will be considered for use in guiding the interpretation. At this point TEIRESIAS chooses a model which 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 area of the investment should be high-technology."

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 INVESTMENT-AREA-IS model), notices that there is no model for INVESTMENT-AREA-IS-HIGH-TECHNOLOGY at the next level, and settles for the former.[6]

7.3.4. *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, and is described here. The second is as a means of allowing TEIRESIAS to see whether the new rule "fits in" to its current model of the knowledge base, and is described in Section 7.5.

To see how the rule models are used to guide the interpretation of the text of the new rule, consider the second line of text typed by the expert. 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. We then consider the possible connotations that each word may have (Fig. 8a). 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.[7]

Code generation is accomplished via a "fill-in-the-blank" mechanism. Associated with each predicate function is a *template*, a list structure that resembles a simplified procedure declaration, and gives the order and generic type of each argument to a call of that function (Fig. 8b). Associated with each of the primitives that make up a template (e.g., ATTRIBUTE, VALUE, etc.) 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; see [5] for details), 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

[6] This technique is used in several places throughout the knowledge transfer process, and in general supplies the model which 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 which concluded about investment area, 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 [5] for an example.

[7] 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. For instance, one of the primitives—the attribute TREND—has associated with it the phrase *the general trend.* Hence when the English word *trends* is found in the text of the rule, it is first changed to *trend* by the canonicalization process, then the connotation pointers are checked, yielding the attribute TREND.

It is possible to have sets of interpretations other than the one shown and TEIRESIAS considers them all. The number of possibilities is kept constrained by enforcing several types of consistency This and other details are omitted here for the sake of brevity; see [5] for a complete description.

connotations in Fig. 8a produces the LISP code in Fig. 8c. The ATTRIBUTE routine finds the attribute TREND, the VALUE routine finds an appropriate value (UPWARD), and the OBJect routine finds the corresponding object type (MARKET) (but following the convention noted earlier, returns the variable name OBJCT to be used in the actual code).

```
THE CLIENT IS FOLLOWING UP ON MARKET TRENDS CAREFULLY
              ↓         ↓     ↓      ↓
        PREDICATE FUNCTION  VALUE  OBJ  ATTRIBUTE
```

FIG. 8a. Connotations.

```
FUNCTION    TEMPLATE
SAME        (OBJ ATTRIBUTE VALUE)
```

FIG. 8b. Template for the predicate function SAME.

```
(SAME OBJCT TREND UPWARD)
The general trend of the market is upward
```

FIG. 8c. The resulting code.

There are several points to note here. First, the interpretation in Fig. 8c is incorrect (the system has been misled by the idiom "following up"); we'll see in a moment how it is corrected. Second, there are typically several plausible (syntactically valid) interpretations 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 (details in [5]). 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 third line of the premise description in Fig. 7 "predicts" that a rule about investment area will contain the attribute *market trend*, and the clause generated from the connotations in Fig. 8a 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 score, the system tends to "hear what it expects to hear" (and that leads it astray in this case).

7.3.5. *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, written in a semi-formal technical language, rather than with large amounts of text in unrestricted dialog written in unconstrained English. Even so, the problem of interpretation is substantial. TEIRESIAS's performance is based on both the application of the ideas noted in Section 7.1 (notably the ideas of building expectations and model-based understanding) and 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 where 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 direction of processing is in general capable of eliminating all ambiguity and finding the correct answer. By moving both top-down and bottom-up, we make use of all available sources of information, resulting in a far more focussed search for the answer. This technique is applicable across a range of different interpretation problems, including 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 [5] 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 [15]).

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 focussing 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.

7.4. Phase III: modifying the interpretation

TEIRESIAS has a simple rule editor that allows the expert to modify existing rules or (as in this example) indicate changes to the system's attempts to understand a new

rule. 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 re-evaluates 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 this case this succeeds in choosing the correct clause next (Fig. 8d shows the correct connotations and resulting code).

```
THE CLIENT  IS  FOLLOWING  UP ON MARKET TRENDS  CAREFULLY
     ↓      ↓       ↓                              ↓
    OBJ   SAME  ATTRIBUTE                        VALUE

(SAME OBJCT FOLLOWS CAREFULLY)
The client follows the market carefully
```

FIG. 8d. The correct interpretation.

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 2 means clause 3 must be spurious, since it attempts to use the word *carefully* in a different sense.

7.5. Phase IV: "second guessing", another use of the rule models

After the expert indicates that TEIRESIAS has correctly understood what he said, 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 in" to 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 Fig. 7:

```
((BRACKET SAME) (FOLLOWS NOTSAME SAME)
                (EXPERIENCE SAME) 1.5)
```

This indicates that when the tax BRACKET of the client appears in the premise of a rule of this sort, then how closely he FOLLOWS the market, and how much investment EXPERIENCE he has typically appear as well. Note that the new rule 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 upon topic for the clause was the amount of investment EXPERIENCE of the client, this must be the attribute to use. The rule model suggests which predicate function to use (SAME, since that is the one paired with EXPERIENCE 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, as the answer to question 6 of the consultation, the expert indicated that the amount of experience was **MODERATE**, so TEIRESIAS uses this as the value. The result is a plausible guess, since it ensures that the rule will in fact work for the current case (note the 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" which 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 was that mentioned above of testing to see how the new knowledge "fits in" to 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 for rules might concern the strength of the inference, number of clauses in the premise, etc. Checks used with, say, a procedural encoding might involve the number and type of arguments passed to the procedure, use of global variables, presence of side effects, etc. In that case, for example, we can imagine adding a new procedure to a system which then responds by remarking that "*. . . most procedures that do hash-table insertion also have the side effect of incrementing the variable* NUMBRELEMENTS. *Shall I add the code to do this*?" In general, this "second guessing" process can involve any characteristic which the system may have "noticed" about the particular knowledge representation in use.

Note also that this second use of the rule model is quite different than the first. Where earlier we were concerned about interpreting text and determining what the expert actually said, here the task is to see what he 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 himself, 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, but 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.

7.6. Phase V: final checkout, bookkeeping, rerunning the consultation

Now that 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 investment area, the area mentioned should be high technology, and the conclusion should be affirmative. The premise should not contain any clauses which are sure to fail in the context in which the rule will be invoked. All these are potential sources of error which would make it obvious that the rule will not fix the bug.

The rule in the current example passes all the tests, but note what would have happened if the user had (perhaps accidentally) typed *the client's income tax bracket is* 20%.

The rule now reads:

RULE 383

If (1) the client's income tax bracket is 20%, and
(2) the client follows the market carefully, and
(3) the amount of investment experience is moderate,
then there is evidence (0.8) that the area of the investment should be high-technology.

Sorry, but this won't work. For Fred Smith, clause No. 1 ("the client's income tax bracket is 20%") is sure to fail.
Would you like to change the rule?
++**

(Since the answer to question 3 of the consultation indicated that the client had a taxable income of $40,000, TEIRESIAS is able to determine [by referring to the appropriate tables] that his tax bracket would not be 20%, hence the rule can't succeed). The expert then has the option of either editing the current rule or writing a new one (since the one he wrote may be correct, only inapplicable to the current problem). If he edits it, the tests are run again, until TEIRESIAS is satisfied that there is nothing obviously wrong with the rule.

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 (e.g., in this case it gets added to the list of rules that conclude about INVESTMENT-AREA),[8] and tagging it with information which 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 which prompted its creation, etc.).

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 sub-process, and checks the results to see if all the problems have been repaired.

[8] Note that these tests require the ability to dissect and partially evaluate the rule. The same function template which is used as a pattern for constructing rules is also used as a guide in this dissection and partial evaluation process. See [5] for details.

8. 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.

8.1. "Knowing what you know"

The MYCIN system has the ability to answer simple natural language questions about the knowledge base. In response to a question like "*How do you determine the identity of an organism causing an infection?*", that system will print the relevant rules. But a rule model, as a generalization of an entire class of rules, answers the question, too. Fig. 9 shows one example, taken from the stock market example we have been using.

**HOW DO YOU DECIDE THAT THE UTILITIES GROUP IS AN APPROPRIATE AREA FOR INVESTMENT?

Rules which conclude that an appropriate area for investment is the utilities group generally use one or more of the following pieces of information:

the desired rate of return of the investment
the income-tax bracket of the client
the time-scale of the investment

Furthermore, the following relationships hold:
If a rule mentions the income-tax bracket of the client, it also typically mentions the time scale of the investment as well.

RULE 247, RULE 45, RULE 123, and RULE 206 conclude that an appropriate area for investment is the utilities group. Which of these (if any) would you like to see?
**

FIG. 9. "Knowing what you know."

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.

8.2. *"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, and base it 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 a knowledge of where it is ignorant.

This makes possible a new option for the expert. Instead of adding a rule to fix a bug, he can ask the system what it would like to learn.

– – ** WHAT WOULD YOU LIKE TO KNOW?
Answer
–Y for yes
–N for no, or
–Q to quit this.
There are only 2 rules for concluding that the name of the stock to invest in would be Homestake-Mining.
Would you like to give me a rule for that? [Y, N, Q]
+ + ** N
There are only 2 rules for concluding that the risk class of an investment should be speculative.
Would you like to offer a rule for that? [Y, N, Q] + + ** Y
The new rule will be called RULE 384
If (1)

FIG. 10. "Knowing what you don't know."

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 in [4] on closed vs. 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 an investor'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 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 would 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 in [4]. Roughly restated, this says that "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.[9]

9. 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.

9.1. The interactive transfer of expertise 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 involving problem solving that is entirely or primarily a high level cognitive task, based on a number of distinct, specifiable principles. Consultations in medicine or 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 process play 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 3) which 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 (like mass spectrum analysis [3]) or synthesized ("toy") domains (like the line drawings in [12]), 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.[10]

[9] This is another useful form of meta-level knowledge.

[10] 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, nor indeed 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.

Knowledge acquisition in context appears to offer useful guidance wherever knowledge of the domain is as yet ill-specified, but the context need not be a shortcoming in the knowledge base uncovered during a consultation, as is done 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 base. Earlier than 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 the present example, in which one rule repairs three bugs; the approach will also work if three independent bugs arise in a consultation. But early in knowledge base construction, where large sub-areas 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 [20]).

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.

9.2. TEIRESIAS as a program

Several difficult problems remain unsolved in the current implementation of the program. There is, for instance, the issue of the technique used to generate the rule models. This process could be made more effective even without using a different approach to concept formation. While 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 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 in both 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.

The weakness of the natural language understanding technique presents a substantial barrier to better performance. Once again there are several improvements that could be made to the existing approach (see [5]), but more sophisticated techniques should also be considered (this work is currently underway; see [1]).

There is also the difficult problem of determining the impact of any new or changed rule on the rest of the knowledge base, which we have considered only briefly (see [5]). The difficulty lies in establishing a formal definition of inconsistency for inexact logics, 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 still substantial.

10. Conclusions

The ideas reviewed above each offer some contribution toward achieving the two goals set out at the beginning of this paper: the development of a methodology of knowledge base construction via transfer of expertise, and the creation of an intelligent assistant.

In the near-term they provide a set of tools and ideas to aid in the construction of knowledge-based programs and represent a few 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 him with a specific example of the performance program's operation, and forcing him to be specific in his 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 insure that it will 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 "fit 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 it 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 dialog was 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 in" to 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 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 is also examined and abstracted to form the rule models (see [8] for additional examples).

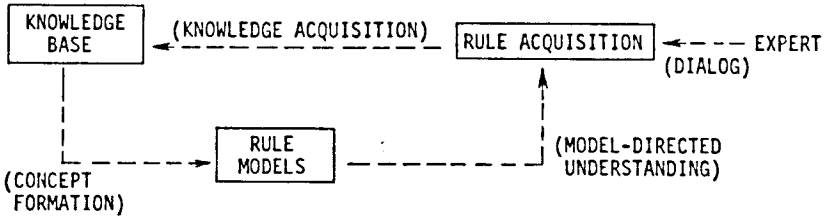


FIG. 11. Model-directed understanding and learning by experience combine to produce a useful feedback loop.

TEIRESIAS also represents a synthesis of the ideas of model-based understanding and learning by experience. While both of these have been developed independently in previous AI research, their combination produces a novel sort of feedback loop (Fig. 11). 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).[11]

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.

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

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 hand-tooled 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.

ACKNOWLEDGMENTS

The work described here was performed as part of a doctoral thesis supervised by Bruce Buchanan, whose assistance and encouragement were important contributions.

REFERENCES

1. Bonnett, A., BAOBAB, a parser for a rule-based system using a semantic grammar, Stanford University HPP Memo 78-10, Stanford CA, U.S.A. (1978).
2. Brown, J. S. and Burton, R. R., Diagnostic models for procedural bugs in mathematical skills, *Cognitive Science* 2 (April–June 1978), pp. 155–192.
3. Buchanan, B. G. and Mitchell, T., Model-directed learning of production rules, in: Waterman and Hayes-Roth (Eds.), *Pattern-Directed Inference Systems* (Academic Press, New York, 1978), pp. 297–312.
4. Carbonell, J. R. and Collins, A. M., Natural semantics in artificial intelligence, Proc. Third International Joint Conference on AI, Stanford, CA (August 1973), pp. 344–351.
5. Davis, R., Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases, Stanford University HPP Memo 76-7 (July 1976).
6. Davis, R., Generalized procedure calling and content-directed invocation, Proc. of the Symposium on Artificial Intelligence and Programming Languages, *SIGART/SIGPLAN* (combined issue, August 1977), pp. 45–54.
7. Davis, R. Knowledge acquisition in rule based systems—knowledge about representations as a basis for system construction and maintenance, in: D. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems* (Academic Press, New York, 1978), pp. 99–134.
8. Davis, R. and Buchanan, B. G., Meta-level knowledge: overview and applications, Proc. Fifth International Joint Conference on AI, Cambridge, MA (August 1977), pp. 920–927.
9. Davis, R., Buchanan, B. and Shortliffe, E. H., Production rules as a representation for a knowledge-based consultation system, *Artificial Intelligence* 8 (February 1977), pp. 15–45.
10. Falk, G., Computer interpretation of imperfect line data, Stanford University AI Memo 132 (August 1970).
11. Feigenbaum, E. A., et al. On generality and problem solving, *Machine Intelligence* 6 (1971), pp. 165–190.
12. Hayes-Roth, F. and McDermott, J., Knowledge acquisition from structural descriptions Proc. Fifth International Joint Conference on AI, Cambridge, MA (1977), pp. 356–362.

13. Mathlab Group, The MACSYMA Reference Manual, MIT Lab. for Computer Science (September 1974).
14. Newell, A. and Simon, H., *Human Problem Solving* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
15. Reddy, D. R., et al. The HEARSAY speech-understanding system: an example of the recognition process, Proc. 3rd IJCAI, Stanford, CA (1973), pp. 185–193.
16. Reiser, J. F., BAIL—A debugger for SAIL, AI Memo 270, Stanford University, AI Lab. (October 1975).
17. Shortliffe, E. H., MYCIN: *Computer-based Consultations in Medical Therapeutics* (American Elsevier, New York, 1976).
18. Shortliffe, E. H. and Buchanan, B. G., A model of inexact reasoning in medicine, *Mathematical Biosciences* 23 (1975), pp. 351–379.
19. Teitelman, W., *The* INTERLISP *Reference Manual*, Xerox Corp. (1975).
20. Waterman, D., Exemplary programming, in: D. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems* (Academic Press, New York 1978), pp. 261–280.
21. Winston, P. H., Learning structural descriptions from examples, Project MAC TR-76, MIT, Cambridge, MA (September 1970).