Report 77-09
Stanford -- KSL

Interactive Transfer of Expertise:
Acquisition of New Inference Rules.

Randall Davis,

Aug 1977

Reprinted from the Proceedings, 5th IJCAI, August 1977.
Used by permission of the International Joint Conference on Artificial Intelligence, Inc.; copies of the Proceedings are available from Morgan Kaufmann Publishers, Inc., 95 First Street, Los Altos, CA 94022, USA.

Interactive Transfe. of Expertise: Acquisition of New Inference Rules

Randall Davis
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

TEIRESIAS is a program designed to function as an assistant in the task of building large, knowledge-based systems. It embodies a particular model of interactive transfer of knowledge from a human expert to the system, and makes possible knowledge transfer 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 an illustration given of its utility and contribution to the creation of intelligent programs.

This work was supported in part by the Bureau of Health Sciences Research and Evaluation of HEW under Grant HS-01544 and by the Advanced Research Projects Agency under ARPA Order 2494. It was carried out on the SUMEX-AIM Computer System, supported by the NIH under Grant RR-00785. The views expressed are solely those of the author.

{1} Introduction

The knowledge base for a high performance, domain-specific program (e.g., DENDRAL [9], MACSYMA[10]) is traditionally assembled by hand, an ongoing task that typically involves numerous man-years of effort. A key element in the construction process 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 chosen this transfer of expertise task as a case study, and have sought to create a program that could supply much the same sort of assistance as that provided by the programmer. That is, we have attempted to create an assistant that will help build intelligent programs. The result is a system called TEIRESIAS [3,5-7], 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.

Work on TEIRESIAS has two 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 assistant capable of offering increasingly more sophisticated aid. This 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 of a specific task (acquisition of new inference rules[1]), for a specific system (a rule-based computer consultant system modelled after the MYCIN system [13,4].) While the discussion deals with one particular task, system and knowledge representation, it should become clear that the main ideas are applicable to a number of more general issues.

{2} Meta-level knowledge

A central theme that runs through this and related papers ([3,5-7]) 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.

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

One of the aims of creating TEIRESIAS was to provide a vehicle for developing a particular approach to knowledge acquisition. We describe that approach here; Section {9} contains some comments on its likely range of applicability.

We view the interaction between the domain expert and the performance program 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 may uncover a fault in the student's knowledge of the subject, and result in the transfer of information to correct it.

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 formalism* 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 those terms 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 between the expert and the system 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

Figure 1 shows a picture of the sort of performance program that TEIRESIAS is designed to help construct. (The program described here is modelled after the MYCIN system, which provided the context within which TEIRESIAS was 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.

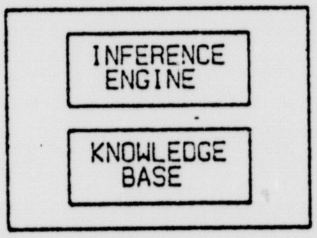

Figure 1 - architecture of the performance program

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. 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). The explicit division also offers a degree of domain independence. 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.

possible to remove the current knowledge base, "plug in" another, and obtain a performance program for a new task.[2]

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 below 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 particular domain of application. The dialog shown is a real example of TEIRESIAS in action that has been transferred to the new domain by substituting a few words 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

Knowledge in the knowledge base is in the form of a collection of decision rules of the sort shown in Figure 2. (The rule is stored internally in the INTERLISP form shown; the English version is generated from that.) 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.

```
RULE027
-------
If [1] the time scale of the investment is long-term, and
   [2] the desired return on the investment is greater than 10%, and
   [3] the area of the investment is not known,
 then AT&T is a likely (.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 .4)
```

Figure 2 - example of a rule

The rules are judgmental, i.e., they make inexact inferences. In the case of the rule in Figure 2, for instance, the evidence cited in the premise is enough to assert the conclusion shown with only a weak degree of confidence (.4 out of 1.0). These numbers embody a model of confirmation described in detail in [14].

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 Figure 2, for example, the predicate function is SAME, and the triple is "*timescale* of *investment* is *long-term*." (The identifier OBJCT is used (rather than the name of a specific object) for reasons dealing with implementation conventions; see [3] for details.)

{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. Assume that the program is attempting to determine which stock would make a good investment. It retrieves all the 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 rule in Figure 2, 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.)

{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 pre-formed, or manufactured by filling in templates with appropriate words (as evidenced by the occasional appearance of phrases like "a area"). Where answers are free text (rather than multiple choice), the system relies primarily on a simple keyword-oriented approach, described below. 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. Dialogues are thus typically not as long as this one.

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 {*italicized inside braces*}. The example is also divided into five phases for later reference.

```
****************************************************************************************

Investment Advisory Program

    1) Investor's name?
    **FRED SMITH

    2) Age?
    **34

    3) Present income?
    **25,000

    4) Number of dependents?
    **3

    5) Occupation?
    **ELECTRICIAN

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

Are all these correct?
++** N
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 [3] to track down the problem. In the second, TEIRESIAS *uses a simple algorithmic search for the error. 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:

RULE027
-------
Since [1.1] the area of the investment is not known,
      [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 (.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?
++** 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 RULE027 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 investment area.
Should it have been possible to conclude a 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:

{At this point TEIRESIAS *displays rules 50 and 166 (omitted here), which would have made the desired conclusion, but were not invoked because one of their premise clauses failed.}*

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 premise clauses that failed was 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 involved:

```
== RULE116 ==>  |
                |→ AREA
== RULE050 ==>  |  RETURN      === RULE027 ===> AT&T
                   TIME SCALE
```

AT&T appeared because RULE027 executed successfully; this occurred in part because investment area was unknown. Investment area was unknown because all rules to conclude it (50 and 116) failed.

The expert believes all existing rules are correct, so 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.*

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 RULE383
 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 (.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:
RULE383
-------
  If    1) The client's income-tax bracket is 50%,
        2) The market has followed a upward trend recently,
        3) The client manages his assets carefully
  Then    there is evidence (.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 occasionally 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 #3).*

This results primarily from the system's simple word-by-word approach to meaning. While the result is in part incorrect, we'll see 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?
++** 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 #2 is correct, then clause #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:
RULE383
-------
  If:  1) the client's income-tax bracket is 50%, and
       2) the client follows the market carefully,
  Then:  there is evidence (.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 the system understands the rule as he entered it, the system checks several things about the rule to see if* _*it*_ *is satisfied.*}

```
**********
{PHASE IV: "second guessing"}
**********

I hate to criticize, Randy, but did you know that most rules about what the
area of a investment might be, that mention-
  the income-tax bracket of the client, and
  how closely the client follows the market
ALSO 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 by relying on the fact that it must be possible to invoke the new rule in the current situation (in order to fix the known bug). 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:
RULE383
-------
  If:  1) the income-tax bracket of the client is 50%,
       2) the client follows the market carefully,
       3) the amount of investment experience of the client is moderate
  Then:  there is evidence (.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;
++**THIS ALLOWED RULE027 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 tagged with a range of such information.}

```
RULE383 has now been added to the knowledge base.
The consultation will be re-run 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.

{Three errors have disappeared with the addition of one rule. This is explained schematically below:

```
  RULE116 *
  RULE050 *
  RULE383  ==> AREA == RULE027 * ==> AT&T
                    == RULE084   ==> DATA-GENERAL
                                     DIGITAL-EQUIPMENT
```

*As before, 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 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 383 supplies one, 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.

I. 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 claim 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, <u>what is it that you know and the system doesn't</u> that allows you to avoid making that same mistake?*

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

II. 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.

III. 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. [3]

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.

IV. Giving programs a model of their 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.

V. 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 [3] for details). This succeeds 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 given the 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.

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.

{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 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.

{7.3.1} **Models and model-based understanding**

As a background for reviewing 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 [8] 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.

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? Not surprisingly, there are regularities in the knowledge base: rules about a single topic tend to have characteristics in common. From these regularities we have constructed *rule models*. These are abstract descriptions of subsets of rules, built from empiricial generalizations about those rules, and are used to characterize a "typical" member of the subset.

Rule models are composed of four parts. 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 implemented 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). [4]

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, etc. But the main idea of characterization is widely applicable and not restricted to any particular representational formalism.

The two other 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. 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.

Rather than being hand-tooled, the models are 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 it forms itself.

The rule models are the primary example of meta-level knowledge in this paper. This form of knowledge, and its generation by the system itself have several implications illustrated in later sections.

Figure 3 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 [3] 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 contains the two kinds of entries noted: 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 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 3.83 (see [3] for precise definitions 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 a indication of reliability.

```
INVESTMENT-AREA-IS

   EXAMPLES  ((RULE116 .33)
              (RULE050 .70)
              (RULE037 .80)
              (RULE095 .90)
              (RULE152 1.0)
              (RULE140 1.0))

   DESCRIPTION
     PREMISE ((RETURNRATE SAME NOTSAME 3.83)
              (TIMESCALE SAME NOTSAME 3.83)
              (TREND SAME 2.83)

              ((RETURNRATE SAME) (TIMESCALE SAME) 3.83)
              ((TIMESCALE SAME) (RETURNRATE SAME) 3.83)
              ((BRACKET SAME) (FOLLOWS SAME) (EXPERIENCE SAME) 1.50))

     ACTION  ((INVESTMENT-AREA CONCLUDE 4.73)
              (RISK CONCLUDE 4.05)

              ((INVESTMENT-AREA CONCLUDE)(RISK CONCLUDE)4.73))

   MORE-GENL  (INVESTMENT-AREA)

   MORE-SPEC  (INVESTMENT-AREA-IS-UTILITIES)
```

Figure 3 - example of a rule model

{7.3.3} Choosing a model

It was noted earlier that the debugging process 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 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 INVESTMENT-AREA-IS-HIGH-TECHNOLOGY model at the next level, and settles for the former.

{7.3.4} **Using the rule model: guiding language interpretation**

The rule models are used in two different ways in the acquisition process. The first is as an aid in understanding the text typed by the expert. (The description below omits many details for the sake of brevity. See [3] for a complete description.)

To see how this works, consider the second line of text typed by the expert: each word may have a number of connotations (Figure 4a). Here connotation means the word might be referring to one or more of the primitives from which rules are built (i.e., a predicate function, attribute, object, or value). One such set of connotations is shown.

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 (Figure 4b). 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 (see [3] 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 connotations in Figure 4a produces the LISP code in Figure 4c. The ATTRIBUTE routine finds the attribute "*market* 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
           ↓                ↓    ↓      ↓
          SAME            VALUE  OBJ  ATTRIBUTE
```

Figure 4a - connotations

FUNCTION	TEMPLATE
SAME	(OBJ ATTRIBUTE VALUE)

Figure 4b - template for the predicate function SAME

```
(SAME OBJCT TREND UPWARD)
```

Figure 4c - the resulting code

There are several points to note here. First, of course, the interpretation in Figure 4c is incorrect; 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 [3]). 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 Figure 3 "predicts" that a rule about investment area will contain the attribute *market trend*, and the clause generated from the connotations in Figure 4a fulfills this prediction. Each interpretation is scored (employing the "strength of advice" number in the rule model) according to how many predictions it fulfills, and this is 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).

Two final comments. First, note that the interpretation process proceeds in what has been called the "recognition" mode [1]: 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 contained in the rule model). Each process contributes to the end result, but it is the combination of them that is effective.

Second, the weakness of the natural language techniques employed here may distract from the impact of the underlying concepts. If the language analysis were more sophisticated, for instance, and could interpret the text correctly a much higher percentage of the time, it might appear that there would be little for the models to do. But this is not so.

Note that more sophisticated analysis routines often rely on an indication of the context in which a dialog is taking place. This, in a primitive fashion, is what the rule models are supplying: the debugging process leads TEIRESIAS to "expect" a certain sort of rule based on what has gone on previously in the conversation, and the particular model chosen offers some advice about what the expert is likely to say. That advice is in turn based on what is already known about how the expert reasons (since it is constructed from rules presently in the knowledge base). Thus even with more sophisticated routines, the models would still have a contribution to make in the interpretation process.

In addition, as will become clear in Section {7.5}, the models can offer a very different sort of advice. While we are dealing here with determining what the expert said, examples there will show how the models can suggest what it is the expert should have said.

{7.4} Phase III: modifying the interpretation

TEIRESIAS has a simple rule editor available 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 (in the belief that the original clause was probably close to correct). In this case this succeeds in choosing the correct clause next (Figure 4d shows the correct connotations and resulting code).

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

            (SAME OBJCT FOLLOWS CAREFULLY)
```

Figure 4d - the correct interpretation

There are also various forms of consistency checking available. One obvious but effective constraint is to insure 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

Now that the expert has indicated that TEIRESIAS has correctly understood what he said, it is the system's turn 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, the presence of a partial match between the new rule and the generalizations in the rule model triggers a response from TEIRESIAS. Recall the last line of the premise description in the rule model of Figure 3:

((BRACKET SAME)(FOLLOWS SAME)(EXPERIENCE SAME) 1.50))

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 insures 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 most general and fundamental point was mentioned above -- testing to see how something "fits in" to the system's 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.

{7.6} Phase V: final checkout

Now that both the expert and TEIRESIAS are satisfied, a final sequence of tests is performed. 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 (see [3] for examples of TEIRESIAS's response if any of these conditions are violated). All these are potential sources of error which 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 (e.g., in this case it gets added to the list of rules that conclude about INVESTMENT-AREA)[5], and tagging it with information which will make it easier to maintain the large and constantly changing body of rules.

At this point, the system also performs any necessary recomputation of rule models, and then reruns the performance program as a sub-process, checking the results to see if all the problems have been repaired.

{8} Other uses for the rule models

Two other uses for the rule models have been developed, illustrating another use of meta-level knowledge. Note that the information in rule models, as a generalization of an entire class of rules, suggests the structure of global trends in the knowledge of the expert who assembled the knowledge base. Thus by simply "reading" the rule model to the user, TEIRESIAS can make clear the overall approach of the system to a given topic (see [3] for an example).

Another use of the models demonstrates that, in a primitive fashion, they give TEIRESIAS a model of what it *doesn't* know. 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. This makes it possible for the expert to ask the system what it would like to learn. In response, the system cycles through the rule models in order, indicating the weakest topics first (see [3]). This is 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 it knows about a subject, compared with how much there is to know.[6]

{9} Analysis

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

{9.1} The interactive transfer of expertise approach

The system has not as yet been tested by having experts use it, but our experience with manual knowledge acquisition provides a perspective on its likely area of greatest utility. 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 implies certain constraints on the range of applicability of this system.

Interactive knowledge transfer seems best suited to task domains involving problem solving that is entirely or primarily a high level cognitive task, with a number of distinct, specifiable principles. Medical diagnosis seems an appropriate domain, but the technique would not seem well suited to those parts of, say, speech understanding or scene recognition in which low level processes appear to play a significant role.

Knowledge acquisition in context appears to offer a useful guide wherever knowledge of the domain is as yet ill-specified, but the context need not be a single consultation, as used here. Our recent experience [12] suggests that an effective context is also provided by examining certain subsets of rules in the knowledge base, using them as a framework for specifying new rules.

Finally, the rule-at-a-time approach is perhaps the most limiting factor. The example given earlier works well, of course, because the bug was manufactured by removing a single rule. In general, the approach 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 given 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 [12], or more generally, deal with larger segments of the basic task (as in [15]).

In general then, this approach seems well suited to the later stages of knowledge base construction for systems performing high-level tasks.

{9.2} TEIRESIAS as a program

As with all first generation systems, TEIRESIAS has many rough spots and inconveniences that have to be smoothed out before it can become an effective user-oriented system.

The technique used to generate the rule models, for instance, could be made more effective. 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 of TEIRESIAS's second guessing ability. Alternatively, more sophisticated concept formation techniques might be borrowed from existing work.

There is 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 assumptions 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.

In addition, 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 [3]), but more sophisticated techniques should also be considered.

There is also the difficult problem of determining the impact of any new or changed rule on the rest of the knowledge base. 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. Additional work is needed here.

{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.

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 offered 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 enabled TEIRESIAS to form a number of expectations about the knowledge it was going to acquire, and made possible several checks on the content of that knowledge to insure that it would in fact fix the bug.

In addition, because the system has a *model of its own knowledge*, it was 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, none of the underlying ideas appears to be limited to this task, or to associative triples or rules as a knowledge representation.

The idea of *model-based understanding*, for instance, found a novel application in the fact that TEIRESIAS has a model of both its own knowledge and the information it expects to receive, and uses this to guide acquisition.

The idea of *biasing the set of models* to be considered offered a specific mechanism for the general notion of *program-generated expectations*, and made possible an assistant whose understanding of the dialog was more effective.

TEIRESIAS was 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 "fit in" to what it already knew*. It was the presence of a partial match between the new knowledge and the system's model of its knowledge that prompted it to make a suggestion to the expert.

The concept of meta-level knowledge made possible *multiple uses of the knowledge in the* *system*: information in the knowledge base was not only used directly (during the consultation), but was 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. While both of these have been developed independently in previous AI research, their combination produced a novel sort of feedback loop (Figure 5; broken lines indicate information flow). 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, and this in turn prompts the recomputation of the relevant rule model(s).

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.

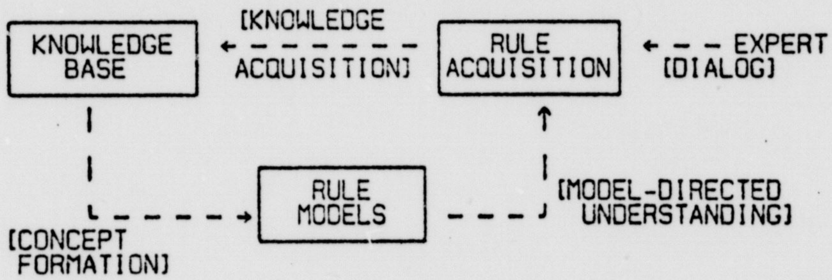

Figure 5

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 at every stage. Nancy Martin, Dave Barstow, and Mike Clancy made a number of very helpful comments on earlier drafts of this paper.

Notes

[1] Acquisition of new conceptual primitives from which rules are built is discussed in [7], while the design and implementation of an explanation capability is discussed in [3] and [11].

[2] 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.

[3] This description is necessarily abbreviated. See [3] for a more detailed discussion.

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

[5] 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 [3] for details.

[6] The issue is related to work described in [2], on closed vs. open sets.

References

[1] Baumgart B G. Geometric models for computer vision, Stanford University AI Memo 249, October 1974.

[2] Carbonell J R, Collins A M. Natural semantics in artificial intelligence, *Proc 3rd IJCAI*, pp 344-351, August 1973.

[3] 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.

[4] Davis R, Buchanan B, Shortliffe E H. Production rules as a representation for a knowledge-based consultation system, *Artificial Intelligence*, 8 (Spring 1977) 15-45.

[5] Davis R. Generalized procedure calling and content directed invocation, to appear in *Proc ACM Conf on AI and Programming Languages*, August 1977.

[6] Davis R, Buchanan B G. Meta-level knowledge: overview and applications, to appear in *Proc 5th IJCAI*, Cambridge, Mass.

[7] Davis R. Knowledge about representations as a basis for system construction and maintenance, to appear in *Pattern-Directed Inference Systems*, Academic Press, (in press)

[8] Falk G. Computer interpretation of imperfect line data, Stanford University AI Memo 132, August 1970.

[9] Feigenbaum E A, et al. On generality and problem solving, in *MI* 6, pp 165-190, Edinburgh University Press, 1971.

[10] Mathlab Group, The MACSYMA Reference Manual, Sept 1974.

[11] Scott A C, Clancey W, Davis R, Shortliffe E H. Explanation capabilities of production-based consultation systems, *Am Jnl Computational Linguistics*, Microfiche 62, 1977.

[12] Scott A C, Aikins J, Buchanan B G, Clancey W, Davis R, van Melle W. Issues of dealing with a growing knowledge base, in preparation.

[13] Shortliffe E H. *MYCIN: Computer-based Consultations in Medical Consultations*, American Elsevier, 1976.

[14] Shortliffe E H, Buchanan B G. A model of inexact reasoning in medicine, *Mathematical Biosciences*, 23 (1975) pp 351-379.

[15] Waterman D, Exemplary programming, in *Pattern-Directed Inference Systems*, Waterman and Hayes-Roth (eds), Academic Press, (in press).

FILMED FROM BEST AVAILABLE COPY