A DIGITALIS THERAPY ADVISOR WITH EXPLANATIONS

William R. Swartout
MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139

Abstract

This paper describes the English explanation facility of the OWL Digitalis Advisor, a program designed to advise physicians regarding digitalis therapy. The program is written in OWL, an English-based computer language being developed at MIT. The system can explain, in English, both the methods it uses and how those methods were applied during a particular session. In addition, the program can explain how it acquires information and tell the user how it deals with that information either in general or during a particular session.

1. Introduction

The documentation of programs (or the lack of it) is a problem that continues to be troublesome. Existing documentation is frequently outdated or inaccurate, can be difficult to obtain, and often can only be comprehended by programmers.

This problem exists for a number of reasons. Documentation is often written only as an after-thought, after a system has been completed. Frequently, the programmer is the only link between the system and its documentation. Thus, changes in the system are not reflected in the documentation unless the programmer remembers or takes the trouble to change it. The documentation is also frequently physically separated from the system, so that a user may not have documentation available when he wishes to use the system. Some programmers try to document the code they produce by using mnemonic names for variables and procedures, yet such documentation remains unavailable to non-programmers.

In this paper, a system is described which can explain itself. This system, called the OWL Digitalis Advisor, is designed to advise physicians concerning digitalis therapy. It is written in OWL I, a prototype of the OWL language currently under development at MIT[Hawkinson, 1975; Long, 1975; Sunguroff, 1976; Martin (in preparation)]. The system is "self-documenting" in the sense that it can produce English explanations of the procedures it uses and the actions it takes directly from code it executes. Most of its explanations are produced in this manner, although a few types of explanation are canned phrases. The physician user may request explanations during a consultation session. The explanations are designed to be understood by a physician with no programming experience.

As Shortliffe [1974] has suggested, if a program can explain its reasoning processes, user acceptance can be more easily obtained, since the user can assure himself that the program makes reasonable deductions which result in reasonable conclusions. Additionally, an explanation facility may serve a valuable pedagogical function. A student or practitioner may use the system to improve his understanding of material he is studying by comparing his own reasoning with that of the system. Finally, the ability to provide explanations serves as a valuable debugging tool.

In the remainder of this introduction, some of the medical aspects of digitalis therapy will be outlined, followed by a review of previous digitalis advisors and work in explanation. Finally, a brief overview of the OWL Digitalis Advisor is given.

1.1 Some Aspects of Digitalis Therapy

In current practice, digitalis is prescribed chiefly to patients who show signs of congestive heart failure and/or conduction disturbances of the heart. Digitalis is useful in treating these conditions, because it increases the contractility of the heart, making it a more effective pump, and slows the conduction of electrical impulses through the conduction system of the heart, thus correcting certain types of arrhythmias. The therapeutic effect of digitalis is achieved by maintaining the level of digitalis in the patient's body within certain bounds which are patient-specific. To determine the correct level for a particular patient, a feedback approach is employed: prescriptions are adjusted based on the effects of prior ones. Although digitalis is a useful and widely used drug [Ogilvie & Reudy, 1972; Doherty, 1973], it is also quite difficult to prescribe correctly. Studies indicate that as many as 20% of all patients receiving digitalis show toxic symptoms, and that the mortality rate among these patients may be as high as 30% [Ogilvie & Reudy, 1972; Peck, Sheiner et al, 1973].

Recently, a digitalis advisor has been developed by Pauker, Silverman and Gorry [1975] which differs from earlier programs [Peck, Sheiner et al, 1973; Jelliffe, Buell, Kalaba et al, 1970; Jelliffe, Buell, Kalaba, 1972; Sheiner, Rosenberg & Melmon, 1972] in two important respects. First, it constructs a patient-specific model, reflecting the program's knowledge of pharmacokinetics and special features of the patient's condition which may alter his response to therapy. This model is used to construct the initial dosage recommendations. Second, the program makes assessments of the toxic and therapeutic effects which actually occur in the particular patient (after he has received the initial dose) to formulate subsequent dosage recommendations, rather than using the blood level of digitalis alone. This program was used as the basis for the OWL Digitalis Advisor.

1.2 Other Work in Explanation

Explanation capabilities have been implemented for systems operating in domains other than digitalis therapy. Winograd's SHRDLU [1972] was an early example of a program with some ability to explain its reasoning. Shortliffe [1974] and Davis [1976] describe the explanation system that has been implemented for MYCIN, a system designed to help doctors in prescribing antibiotics. MYCIN functions in an interactive manner, and is capable of explaining why certain questions were asked, as well as the reasoning chain that it employs either in general or for a particular patient. The explanation systems of MYCIN and the OWL Digitalis Advisor are compared later in this paper.

Mikelsons has been working on the problem of explaining programs written in BDL (Business Definition Language) to a user unfamiliar with programming [Mikelsons, 1975]. His system uses two models: one to model the program's understanding of the problem and the other the user's. Mikelsons' system is still under development, hence it is impossible to compare the performance of his system with that of the OWL Digitalis Advisor. However, it does seem safe to say that his system is different from the Digitalis Advisor. The most fundamental difference is that when the Digitalis Advisor was written, an effort was made

to use one structure as much as possible as a combined user and program model. We will see that in most cases exhibiting an English translation of appropriate portions of this single model to the user is sufficient to give the Digitalis Advisor a good explanatory capability. In those cases where it is not possible to model the user's understanding of the problem with the computational model, the Digitalis Advisor uses comments placed in the code to re-cast the explanation in terms understandable to the user. Thus, while Mikelsons' system will use the more sophisticated (but also more complex) two model approach exclusively, the OWL Digitalis Advisor relies on a simpler single model.

NOAH [Sacerdoti, 1975] uses procedural nets to represent actions at various levels of detail, and can construct new plans from those that exist in the net. The Digitalis Advisor was not designed to construct new plans, but, like NOAH, is structured by levels of abstraction to facilitate the explanation process.

1.3 An Overview of the OWL Digitalis Advisor

The OWL Digitalis Advisor consults with a physician in an interactive manner. The Advisor asks the clinician a number of questions about the patient and then produces a set of recommendations. After the patient has received an initial dose of digitalis, the program can produce a new dosage regimen based on the reaction of the patient.

While a session is taking place, the system can explain why it is asking a question. At the end of a session, the system can provide several distinct types of explanation. It can explain its procedures and actions either in general or for the patient at hand. It can explain how variables in its model are set or used either in general or for a particular patient. The system can offer the above explanations for previous sessions as well as for the current one. The system also allows the user to change his answers to determine the effect of different inputs on the system's recommendations. When an answer is changed, the system recomputes *only* steps that are affected--a process called "revising". Once the affected steps have been recomputed, the system can provide the user with a concise explanation of the effects of the change.

Originally, I had hoped that the Digitalis Advisor would be able to accept requests for explanations in English. Unfortunately, work on the OWL Parser[1] has been delayed. Until it is completed, requests for explanations must be made in the form of LISP function calls.

2. Programming for Explanation

The designer of a system that can explain itself faces a number of problems. One is to provide the user with an explanation that answers his question, yet does not swamp him with details. To accomplish this, the information contained in the system needs to be structured in some way. Different methods for structuring the information have been proposed.

In the MYCIN system, Davis uses the certainty factor of a rule as an indication of its "informational content". Those rules that have a higher certainty factor are said to contain less information, because the designers of MYCIN feel that they are more like definitions. Rules with lower certainty factors (hence less certain conclusions) supposedly contain more information. Davis uses the expression *-log(certainty factor)* as the measure of the information contained in a rule. When the user asks why a particular rule was invoked, he has the ability to specify how much information he wishes to see displayed at once by specifying a number between 1 and 10. The system goes up the goal path from the rule in question, explaining rules until the sum of the measures of information in the rules explained so far divided by the sum of the measures of information in all the rules between the rule in question and the top goal is equal to the number supplied by the user divided by 10. If the user specifies a high number, many goals will be displayed together, while a low number will cause just a few goals above the rule in question to be displayed. Since the number of rules explained is based on their information content, for a given number, rules with high certainty factors will tend to be explained in conjunction with other rules, while those with low certainty factors will be explained alone. Using this method has the desirable effect that the uncertain actions that the system takes will tend to be highlighted (since they will tend to be explained alone) while definitional rules will be explained in conjunction with other rules. The problem with this scheme is that the certainty with which a conclusion is reached does not necessarily indicate how important it is to explain the conclusion. In domains where conclusions may be reached with a high degree of certainty, the above method would indicate that it was just as important to explain one conclusion as any other. In the Digitalis Advisor, it is certain that a patient must be checked for digitalis sensitivities, yet it is also a very relevant thing to explain to a physician. In designing the Digitalis Advisor, an attempt was made to make the structure of the program reflect the structure of the methods used by doctors in prescribing digitalis.

Designers of rule-based systems have also had trouble expressing knowledge about actions in the rule format. Davis [1976, page 29] notes:

> "A ... problem is the limit on the amount of knowledge which can conveniently be expressed in a single rule. Actions which are "larger" than this limit are often achieved by the combined effects of several rules. For many reasons, this is difficult to do, and often produces opaque results."

Davis [1976, page 261] also observes:

> "Rules are a reasonably natural and convenient form of knowledge encoding for what may be termed "single level" phenomena - it is easy to think of single decisions or actions in terms of a rule.
>
> Experience with MYCIN has demonstrated, however, that even experts acquainted with the program tend to think of a sequence of operations in procedural terms, and find flowcharts the most convenient medium of expression. While the flowcharts can always be converted to an equivalent set of rules, the conversion is non-trivial, and sometimes requires reconsidering the knowledge being expressed, since the two methodologies offer different perspectives on knowledge organization and use.

In designing the OWL Digitalis Advisor, it was decided to use a procedural system so that knowledge of actions could be placed in a hierarchical structure of procedures. Since it is possible to group knowledge conveniently in this way, we will see that the explanations produced by the OWL Digitalis Advisor are well-structured.

Another problem that confronts the system designer is the problem of reconciling the user's model of the problem with the program's model of the problem. That is, when explaining the program to the user, it is necessary to take into account the possibility that the user's model of the problem is very different from the program's. Mikelsons has proposed the use of two models. A difficultly with this approach is that when the system

1. A part of the OWL system which converts natural English sentences into OWL forms.

is modified, changes must be made not only to the program, but to the structures linking it to the user's model as well. It seems that to avoid the dangers of unintentional discrepancies between the models it would be a good idea to incorporate the user's model into the actual program as much as possible.

The Digitalis Advisor is structured into what are often called "levels of abstraction". The basic idea behind this approach is that the higher level procedures in the structure represent more general goals or actions, and these high level procedures call more specific procedures (which in turn call still more specific procedures). As mentioned above, this approach was used in Sacerdoti's NOAH system. The idea of levels of abstraction also closely parallels ideas developed in structured programming. In his "Notes on Structured Programming" [1972, page 44] Dijkstra states:

> If I judge a program by itself, my central theme, I think, is that I want the program written down as I can understand it, I want it written down as I would like to explain it to someone.

In the Digitalis Advisor, the procedure used to start treating a patient with digitalis is called *BEGIN THERAPY*, written [(BEGIN THERAPY)] in OWL I. One of the functions that [(BEGIN THERAPY)] calls is a procedure that checks for any sensitivities the patient may have. It is called [(CHECK SENSITIVITIES)]. [(CHECK SENSITIVITIES)], in turn, calls a number of subprocedures. One of these is [(CHECK (SENSITIVITY (DUE (TO POTASSIUM)))))], which checks for digitalis sensitivity due to a potassium imbalance. When the method or event[2] for beginning therapy is described, [(CHECK SENSITIVITIES)] is displayed to the user as ***CHECK SENSITIVITIES*** without any of the structure beneath it. It summarizes the calls below it, so that they do not have to be displayed. If the user is curious about how sensitivities are checked, he may ask, and he will see that one of the steps is to check sensitivity due to potassium. If he is still curious, he may inquire about the details of that step as well. Notice that if he is not interested in any details, the entire process of checking sensitivities will be summarized as one step, so that he will not get output that he does not care about.

When a method is explained, in the current implementation of the Digitalis Advisor, it is assumed that a call to another method can be taken as a summary of the actions performed by that plan. Thus, only the call need be displayed. The event created by the call is not offered as part of the explanation unless the user specifically asks about it. In the future, it might be desirable if the call could be flagged to indicate exceptions to this convention.

Notice that this method of summarizing output contrasts with the certainty factor approach adopted in MYCIN. Rather than attempting to make conclusions about the information content of a rule based on its certainty factor, we are attempting to structure the procedures of the Advisor so that they model the structure of the expert's solution. This methodology places a burden on the system designer since he is no longer free to structure the program in any manner, but instead must attempt to model the expert's methods with it. It also assumes that the user will be able to understand explanations derived from such a program.

2. Methods in OWL I roughly correspond to procedures in other languages. Events are trace structures of the execution of a plan, left behind by the OWL Interpreter as it runs. Calls in OWL are like calls in other languages, except that methods are invoked by a pattern match.

2.1 The Explanation Routines -- How They Work

In this section, the various explanation routines provided by the Digitalis Advisor are outlined. The explanation routines produce explanations directly from the OWL I code the interpreter runs, and from the "event structure" the interpreter creates. The explanations are not canned -- a change in the procedures used by the Advisor will be reflected in changed explanations. Even though they can translate the OWL I methods to English, the explanation routines are quite simple. Simplicity is possible because the OWL I code itself is close to English and because the Advisor is structured into levels of abstraction.

2.1.1 Describing Methods

One of the simplest explanation routines is DESCRIBE-METHOD, which describes OWL I methods. This procedure is designed to answer the question "In general, how do you _____?". DESCRIBE-METHOD describes how an OWL I procedure works in general, not how it applies to a particular patient. The routine is called with a single argument, which is the OWL I plan to be described. DESCRIBE-METHOD traces out the links which connect the steps of the OWL method and converts the steps to English as it encounters them. Special routines are called recursively to explain certain OWL primitives such as BECOME, IF-THEN, and OR[3]. If DESCRIBE-METHOD encounters a call to another OWL I plan, it only displays that call. As it produces an explanation, the system indents the output to indicate the structure of the OWL plan. As an example, an OWL method is listed below, followed by its English explanation.

```
[(CHECK (SENSITIVITY (DUE (TO THYROID-FUNCTION))))
 SUMMARY: (FACTOR (REDUCTION (DUE (TO MYXEDEMA))))
 METHOD:
  (IF-THEN
     (CURRENT-VAL (STATUS MYXEDEMA) UNKNOWN)
     (ASK-USER (QUANTA T4)):),
    (OR
     (IF-THEN
      (OR:15
       (STATUS MYXEDEMA PRESENT)
       (AND:10
        (STATUS MYXEDEMA UNKNOWN)
        (LESS-THAN 2.5 (QUANTA T4))))
      (BECOME-ALSO
       (CONDITIONS
          CORRECTABLE-AND-PRESENT MYXEDEMA)):1,
      (UNBECOME (CONDITIONS DEGRADABLE MYXEDEMA)):1,
      (BECOME
       (FACTOR
        (REDUCTION (DUE (TO MYXEDEMA))) 0.67)):1,
      (BECOME-ALSO (REASONS REDUCTION MYXEDEMA)):)
     (AND
      (BECOME-ALSO
       (CONDITIONS DEGRADABLE MYXEDEMA)):2
      (UNBECOME
       (CONDITIONS
          CORRECTABLE-AND-PRESENT MYXEDEMA)):2
      (BECOME
       (FACTOR
        (REDUCTION (DUE (TO MYXEDEMA))) 1.0)):2
      (UNBECOME (REASONS REDUCTION MYXEDEMA)):)))]
```

The OWL Code to Check for Sensitivity Due to Myxedema

3. BECOME statements assert facts, IF-THENs are conditionals, and ORs correspond to COND statements in LISP.

```
(DESCRIBE-METHOD
   [(CHECK
      (SENSITIVITY (DUE (TO THYROID-FUNCTION)))))])
```

TO CHECK SENSITIVITY DUE TO THYROID-FUNCTION I DO THE FOLLOWING STEPS:

1. IF THE CURRENT VALUE OF THE STATUS OF MYXEDEMA IS UNKNOWN THEN I ASK THE USER THE LEVEL OF T4.

2. I DO ONE OF THE FOLLOWING:

2.1 IF EITHER THE STATUS OF MYXEDEMA IS PRESENT OR THE STATUS OF MYXEDEMA IS UNKNOWN AND THE LEVEL OF T4 IS LESS THAN 2.50 THEN I DO THE FOLLOWING SUBSTEPS:

2.1.1 I ADD MYXEDEMA TO THE PRESENT AND CORRECTABLE CONDITIONS.

2.1.2 I REMOVE MYXEDEMA FROM THE DEGRADABLE CONDITIONS.

2.1.3 I SET THE FACTOR OF REDUCTION DUE TO MYXEDEMA TO 0.67.

2.1.4 I ADD MYXEDEMA TO THE REASONS OF REDUCTION.

2.2 OTHERWISE, I ADD MYXEDEMA TO THE DEGRADABLE CONDITIONS, REMOVE MYXEDEMA FROM THE PRESENT AND CORRECTABLE CONDITIONS, SET THE FACTOR OF REDUCTION DUE TO MYXEDEMA TO 1.00 AND REMOVE MYXEDEMA FROM THE REASONS OF REDUCTION.

An English Explanation of the Code to Check Sensitivity Due to Myxedema

2.1.2 Describing Events

The explanation routine which describes events is called DESCRIBE-EVENT. It is designed to answer the question "For this patient, how did you _____?". This routine is a little more sophisticated, since a certain amount of editing must be done to avoid making nonsensical explanations. The principal difference between explaining events and explaining methods is that when methods are explained, all possible paths through the method are outlined, but when events are explained, only the specific path taken during the event is displayed. Thus, as one would expect, the chief differences between DESCRIBE-METHOD and DESCRIBE-EVENT are to be found in the routines that explain conditional statements.

When a simple conditional statement is encountered while explaining an event, a check is made to see if the predicate of the conditional succeeded or failed. The OWL event structure contains this information. If the predicate failed, the statement is normally not described. If the predicate succeeded, the predicate is given as the reason for the actions taken by the statement. Sufficient information is stored away by the interpreter as it executes so that it is always possible to give the actual reason for a decision, even in the case of conjuncts and disjuncts.

The OWL OR statement, which corresponds to the COND statement in LISP, is a more complex case. The OR statement may contain several conditional statements and it can serve two distinct purposes. On the one hand, it can be used like a CASE statement, that is, each of the clauses of the OR may involve the same variable, and all of the clauses together cover a set of disjoint possibilities. In that case, the order of the clauses usually does not matter, and the most appropriate explanation is merely to give the predicate that succeeded as the reason for the action taken. On the other hand, each of the clauses of the OR may involve a different variable. In that case, the ordering of the clauses is often important, and it seems that in explaining the OR, the predicates that failed as well as the one that succeeded should be given as reasons for the actions taken by the statement. To determine the type of the OR statement, the explanation routine examines the variables used in the predicates before explaining the statement. In future versions of the OWL I interpreter, it might be a good idea to use two different types of statements to eliminate the ambiguity.

To make explanations of numerical computations clearer, the value of a numeric variable is printed in parentheses following the variable whenever it is displayed. The values of non-numeric variables are usually clear from the context of the explanation and are not specifically displayed unless an assertion about the variable is being described. Whenever a new assertion is made, the new value and the old value of the variable are both given. A sample explanation of an event is reproduced below.

```
(DESCRIBE [(CHECK (SENSITIVITY (DUE (TO THYROID-
FUNCTION)))))])
```

DO YOU ONLY WANT TO SEE EVENTS FROM THE CURRENT SESSION? (YES OR NO) n

DURING THE SESSION ON 9/21/76 AT 11:10, I CHECKED SENSITIVITY DUE TO THYROID-FUNCTION BY EXECUTING THE FOLLOWING STEPS:

1. I ASKED THE USER THE STATUS OF MYXEDEMA. THE USER RESPONDED THAT THE STATUS OF MYXEDEMA WAS PRESENT.

2. SINCE THE STATUS OF MYXEDEMA WAS PRESENT I DID THE FOLLOWING:

2.1 I ADDED MYXEDEMA TO THE PRESENT AND CORRECTABLE CONDITIONS. THE PRESENT AND CORRECTABLE CONDITIONS THEN BECAME MYXEDEMA.

2.2 I REMOVED MYXEDEMA FROM THE DEGRADABLE CONDITIONS. THE DEGRADABLE CONDITIONS THEN BECAME HYPOKALEMIA, HYPOXEMIA, CARDIOMYOPATHIES-MI, AND POTENTIAL POTASSIUM LOSS DUE TO DIURETICS.

2.3 I SET THE FACTOR OF REDUCTION DUE TO MYXEDEMA TO 0.67. THE FACTOR OF REDUCTION DUE TO MYXEDEMA WAS PREVIOUSLY UNDETERMINED.

2.4 I ADDED MYXEDEMA TO THE REASONS OF REDUCTION. THE REASONS OF REDUCTION THEN BECAME MYXEDEMA.

An Explanation of the Event of Checking for Sensitivity Due to Myxedema

2.1.3 Explaining the Use and Setting of Variables

Since the OWL Knowledge Base is completely cross-referenced, the Digitalis Advisor can explain how program variables are set and used. A description of the functions that perform this task, together with examples of their use may be found in Swartout [1977].

2.2 Dealing with Unfamiliar Methods

When writing a computer program, it is sometimes necessary to use methods that are totally foreign to users of the system. This may be because the methods employed by humans are unknown, too inefficient or otherwise inappropriate for computer implementation. Whenever this situation occurs, it will not be possible to give meaningful explanations by merely translating the code of the program into English. For example, the use of a weighted-sum to determine the clinical condition of a patient may be quite foreign to the average doctor.

To deal with this problem in the Digitalis Advisor, we have attached English comments to the OWL code in those few places where the methods employed are not familiar to physicians. When the OWL method is explained, the comments are displayed along with the translated OWL code. The comments are intended to make the translated code understandable to the user. When an event is displayed, only comments associated with steps that actually executed are displayed together with the steps.

Although this simple solution is adequate for the Digitalis Advisor, it would probably not be sufficient if the methods used by the program and user were different in complex ways. If the method used by humans were too inefficient, it might be best to use two methods: one for machine execution, and the other for explanation. This approach is similar to Mikelsons'.

2.3 Revising

A question which may occur to the user of an expert consulting program is: "What would happen to the program's recommendations if I revised this answer I have given?". How the changes resulting from such revisions can be explained to the user will depend in part on how the recommendations are recomputed. In MYCIN [Shortliffe, 1974], new recommendations are produced by accepting the changed answer and then recomputing the entire session. This approach has the disadvantage that many steps which are not affected by the change will be re-executed. Thus, when an explanation of a revised recommendation is given to the user, either a large number of steps irrelevant to his question will be explained, or a rather sophisticated explanation module must be built to eliminate the explanation of unaffected steps.

Another approach, used by the Digitalis Advisor, is to use a smarter interpreter that will re-execute only those steps that could possibly need it[4]. This approach has the advantages that the re-execution process is generally more efficient and that the simple explanation routines described above (with a few minor modifications) can be used to produce concise explanations of the effects of revising an answer. A sample revision is given below.

This revision occurs at the end of the sample session given in the appendix. The user indicates to the system that he wishes to make a new hypothesis about the status of hypoxemia.

YOU STATED THAT THE STATUS OF HYPOXEMIA WAS NOT SUSPECTED. IF YOU WISH TO CHANGE THAT ANSWER, PLEASE GIVE YOUR NEW RESPONSE BELOW.

WHAT IS THE STATUS OF HYPOXEMIA? ====> suspected

The system reminds the user of his previous response and asks him for a new one. The user indicates that he suspects hypoxemia. This answer is different than the one he gave during the original session.

WHAT IS THE LEVEL OF PO2? ====> 45

PLEASE TRY TO CORRECT THE HYPOXEMIA AS SOON AS POSSIBLE.

The user indicates that the patient is quite hypoxemic, and the advisor suggests correcting the condition. Since the change in the status of hypoxemia has resulted in a change in the recommendations, the advisor prints the new parameters and makes a new prescription.

PARAMETERS USED TO MAKE RECOMMENDATIONS:
BODY STORE GOAL ===> 0.371 MG
PROJECTED AMOUNT ON BOARD ===> 0.000 MG
BODY STORES SHIFT ===> 0.371 MG
REMAINING TIME TO REACH GOAL ===> 48.0 HRS.
HALF LIFE ===> 43.7 HOURS 1.8 DAYS
DAILY LOSS AT EQUILIBRIUM ===> 0.117 MG
ABSORPTION FACTOR ===>0.75

Note that the recommended body stores goal was 0.55 mg before the revision.

THE DOSAGE RECOMMENDATIONS ARE:

HOURS FROM NOW	ORAL	IV
NOW	125 + .0625 MG	.125 MG

REPORT BACK AFTER THE FIRST DOSE.

8	.125 MG	.0625 MG
16	.0625 MG	.125 MG
24	.125 MG	.0625 MG
32	.0625 MG	.0625 MG
40	.0625 MG	.0625 MG

ORAL MAINTENANCE SCHEDULE: ALTERNATE .25 MG & .125 MG

IV MAINTENANCE SCHEDULE: .125 MG
REVISION COMPLETED.

The system now returns to top level, indicating that all necessary changes have been made. The user may now ask the system to describe the revision.

(describe-revision)

THE RESULTS OF THE REVISION WERE:

1. WHILE CHECKING SENSITIVITY DUE TO HYPOXEMIA, I ASKED THE USER THE STATUS OF HYPOXEMIA. I MADE A DIFFERENT DECISION THAN BEFORE THE REVISION. SINCE THE CURRENT VALUE OF THE STATUS OF HYPOXEMIA WAS SUSPECTED I DID THE FOLLOWING:

    1.1 I ASKED THE USER THE LEVEL OF PO2. THE USER RESPONDED THAT THE LEVEL OF PO2 WAS 45.

    1.2 SINCE THE LEVEL OF PO2 (45) WAS LESS THAN 50.00 I DID THE FOLLOWING:

        1.2.1 I SET THE FACTOR OF REDUCTION DUE TO

4. The algorithm used by the interpreter to accept revisions is beyond the scope of this paper. A more complete description of it, a proof of its correctness, and approaches proposed by others may be found in Swartout [1977], where it is called updating.

HYPOXEMIA TO 0.67. (BEFORE REVISING, THE VALUE WAS 1.00.)

1.2.2 I ADDED HYPOXEMIA TO THE PRESENT AND CORRECTABLE CONDITIONS. THE PRESENT AND CORRECTABLE CONDITIONS BECAME HYPOXEMIA AND MYXEDEMA.

1.2.3 I REMOVED HYPOXEMIA FROM THE DEGRADABLE CONDITIONS. THE DEGRADABLE CONDITIONS BECAME HYPOKALEMIA, CARDIOMYOPATHIES-MI, AND POTENTIAL POTASSIUM LOSS DUE TO DIURETICS.

1.2.4 I ADDED HYPOXEMIA TO THE REASONS OF REDUCTION. THE REASONS OF REDUCTION BECAME HYPOXEMIA AND MYXEDEMA.

2. I MADE A DIFFERENT DECISION THAN BEFORE THE REVISION. SINCE THE STATUS OF HYPOXEMIA WAS SUSPECTED AND THE LEVEL OF PO2 (45, FORMERLY UNDETERMINED) WAS LESS THAN 65.00 I SUGGESTED CORRECTING HYPOXEMIA.

3. WHILE COMPUTING THE FACTOR OF ALTERATION, I SET THE FACTOR OF ALTERATION DUE TO SENSITIVITIES TO THE PRODUCT OF THE FACTOR OF REDUCTION DUE TO ADVANCED AGE (1.00), THE FACTOR OF REDUCTION DUE TO HYPERCALCEMIA (1.00), THE FACTOR OF REDUCTION DUE TO HYPOKALEMIA (1.00), THE FACTOR OF REDUCTION DUE TO POTENTIAL POTASSIUM LOSS DUE TO DIURETICS (1.00), THE FACTOR OF REDUCTION DUE TO HYPOXEMIA (0.67), THE FACTOR OF REDUCTION DUE TO MYXEDEMA (0.67), AND THE FACTOR OF REDUCTION DUE TO CARDIOMYOPATHY-MI (1.00). THE FACTOR OF ALTERATION DUE TO SENSITIVITIES WAS SET TO 0.45. (BEFORE REVISING, THE VALUE WAS 0.67.)

4. I SET THE FACTOR OF ALTERATION TO THE PRODUCT OF THE FACTOR OF ALTERATION DUE TO SENSITIVITIES (0.45) AND THE QUOTIENT OF THE WEIGHT OF THE PATIENT (72) AND 70.00. THE FACTOR OF ALTERATION WAS SET TO 0.46. (BEFORE REVISING, THE VALUE WAS 0.69.)

5. WHILE COMPUTING THE BODY-STORES GOAL, I SET THE BODY-STORES GOAL TO THE PRODUCT OF THE FACTOR OF ALTERATION (0.46) AND THE BASIC BODY-STORES GOAL (0.80). THE BODY-STORES GOAL WAS SET TO 0.37. (BEFORE REVISING, THE VALUE WAS 0.55.)

6. WHILE GIVING RECOMMENDATIONS, I PRINTED THE PARAMETERS.

7. I MADE THE PRESCRIPTION.

3. Limitations and Advantages of this Approach

This paper presents an approach to program explanation based on the code of the program and a trace of its execution. The chief advantage of this approach is its simplicity. If a program can be written in a sufficiently structured, close-to-English style, very little additional work needs to be done to produce explanations of the code. Additionally, if the program is modified, the changes are immediately reflected in the explanations.

Because of this simplicity, this approach is most applicable to those programs which closely model methods employed by humans. If a user is not familiar with the methods employed by the program, he will have trouble understanding the explanations. Thus, while physicians seem to understand the explanations offered by the digitalis advisor, the average layman has trouble if the concepts behind digitalis therapy are not explained to him.

While structuring programs using levels of abstraction is perhaps a better way of indicating the relative importance of steps than the method used by MYCIN, it is not the final answer. For one thing, there is currently no way to indicate the relative importance of steps at the same level. While it seems that it would be possible to solve this problem by placing markers on the steps, this has not yet been done. More importantly, this approach makes the assumption that the importance of a step in terms of explanation is closely modelled by its level of abstraction. While this seems to be true in the Digitalis Advisor, it might not be true for all domains.

Another limitation is that the sorts of explanations that the system may produce are fixed when the program is written. If the explanation produced at one level of abstraction is too general, and the explanation at the next level down is too specific, the user has no way out, since there is no way to produce an explanation at an intermediate level if that level does not exist in the program.

Finally, the system described above has no model of the user or what he is trying to accomplish. Thus, cardiologists and laymen receive the same explanation, just as the explanations are no different for those who are trying to learn from the system than for those who are trying to debug it. As may be seen from some of the above examples, the system may display some things to a physician which should really only be shown to someone maintaining it. We are currently working on incorporating a model of the user into the system.

Despite these limitations, we have found the explanation facility to be very useful in developing the Digitalis Advisor, since program bugs are much more apparent when they are expressed in English, and we have been encouraged to find that doctors who have seen the system generally find the explanations helpful and understandable.

4. A Sample Initial Session

Entries made by the user are in lower case. Comments are in italics. In the actual implementation, most of the questions below are asked in a multiple-choice format, which was not used here to save space.

WHAT IS THE AGE OF THE PATIENT? 56

WHAT IS THE WEIGHT OF THE PATIENT? (IN KG.) 72

WHAT IS THE SEX OF THE PATIENT? male

WHAT IS THE CARDIAC RHYTHM? atrial fibrillation

IS THERE REGULARIZATION OF THE VENTRICULAR RHYTHM? absent

WHAT IS THE VENTRICULAR RATE? 123

WHAT IS THE STATUS OF PVCS? absent

WHAT IS THE REASON OF DIGITALIZATION? arrhythmia

WHAT IS THE STATUS OF DIGITALIS USE? absent

IS THE URINE OUTPUT OF THE PATIENT GREATER THAN 400CC PER DAY? yes

IS THE RENAL FUNCTION STABLE? stable

WHAT SORT OF RENAL FUNCTION MEASURE IS AVAILABLE? creatinine-clearance
ENTER RECENT CREATININE-CLEARANCE VALUES WITH DATE AND TIME.

DATE	TIME	TEST VAL	RENAL F
9/21/76	10:00	100	0.80

WHAT IS THE LEVEL OF SERUM CALCIUM? 7

HAS IV CALCIUM BEEN ADMINISTERED WITHIN THE LAST TWO HOURS? no

WHAT IS THE LEVEL OF SERUM POTASSIUM? 4

IS THE PATIENT BEING GIVEN OR ABOUT TO BE GIVEN DIURETICS? no

DOES THE PATIENT HAVE CARDIOMYOPATHY OR AN ACUTE MI? absent

WHAT IS THE STATUS OF HYPOXEMIA? not suspected

WHAT IS THE STATUS OF MYXEDEMA? present

HOW LONG DO YOU WISH TO TAKE TO REACH FULL DIGITALIZATION? 2 days

Finally, the system produces its recommendations.

PARAMETERS USED TO MAKE RECOMMENDATIONS:
BODY STORE GOAL ===> 0.554 MG
PROJECTED AMOUNT ON BOARD ===> 0.000 MG
BODY STORES SHIFT ===> 0.554 MG
REMAINING TIME TO REACH GOAL ===> 48.0 HRS.
HALF LIFE ===> 43.7 HOURS 1.8 DAYS
DAILY LOSS AT EQUILIBRIUM ===> 0.175 MG
ABSORPTION FACTOR ===>0.75

HOW OFTEN DO YOU WISH TO ADMINISTER DIGITALIS? tid

The system asks the user how often he wishes to give digitalis. The expression "tid" means three times per day.

THE DOSAGE RECOMMENDATIONS ARE:

HOURS FROM NOW	ORAL	IV
NOW	.25 MG	.125 + .0625 MG

REPORT BACK AFTER THE FIRST DOSE.

8	.125 + .0625 MG	.125 MG
16	.125 MG	.125 MG
24	.125 MG	.125 MG
32	.125 MG	.0625 MG
40	.125 MG	.0625 MG

ORAL MAINTENANCE SCHEDULE: .25 MG

IV MAINTENANCE SCHEDULE: ALTERNATE .25 & .125 MG

5. Acknowledgements

I would like to thank all those who made this possible. Howard Silverman and Dr. Stephen Pauker were very helpful with medical assistance. Alex Sunguroff helped with the OWL Interpreter. Various discussions with members of the Automatic Programming Group, the Clinical Decision Making Group, and, in particular, Lowell Hawkinson, provided many valuable insights. I would especially like to thank my supervisors, Professors William A. Martin and Peter Szolovits for their patience and advice.

This research was supported in part by the Health Resources Administration, U. S. Public Health Service, under grant 1 R01 MB 00107-01 from the Bureau of Health Manpower and under grant HS 00911-01 from the National Center for Health Services Research.

6. References

1. Dahl OJ, Dijkstra EW, Hoare CAR: *Structured Programming.* Academic Press, 1972
2. Davis R: Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases. SAIL AIM-283, 1976
3. Doherty JE: Digitalis Glycosides: Pharmacokinetics and their clinical implications. Ann Intern Med 79:229-238, 1973
4. Hawkinson L: The representation of concepts in Owl. Proceedings of the Fourth IJCAI, 1975
5. Jelliffe RW, Buell J, Kalaba R et al: A computer program for digitalis dosage regimens. Math Biosci 9:179-193, 1970
6. Jelliffe RW, Buell J, Kalaba R: Reduction of digitalis toxicity by computer-assisted glycoside dosage regimens. Ann Intern Med 77:891-906, 1972
7. Long W: Question answering in Owl. Automatic Programming Group Internal Memo
8. Martin WA: A theory of English grammar. MIT Laboratory for Computer Science Technical Memo (in preparation)
9. Mikelsons M: Computer assisted application description. Second ACM Symposium on Principles of Programming Languages, 1975
10. Ogilvie RI, Reudy J: An educational program in digitalis therapy. JAMA 222:50-55, 1972
11. Peck CC, Sheiner LB et al: Computer-assisted digoxin therapy. N Eng J Med 289:441-446, 1973
12. Sacerdoti E: A structure for plans and behavior. SRI TN 109, 1975
13. Sheiner LB, Rosenberg B, Melmon K: Modelling of individual pharmacokinetics for computer-aided drug dosage. Computers and Biomedical Research 5:441-459, 1972
14. Shortliffe EH: MYCIN: A rule-based computer program for advising physicians regarding antimicrobial therapy selection. SAIL AIM 251, 1974
15. Silverman H: A digitalis therapy advisor. MAC TR-143, 1975
16. Sunguroff A: OWL interpreter reference manual. MIT Automatic Programming Group Internal Memo, 1976
17. Swartout WR: A digitalis therapy advisor with explanations. MIT Laboratory for Computer Science TR-176, 1977
18. Winograd T: *Understanding Natural Language,* PhD thesis, Academic Press, 1972