5

Details of the Consultation System

Edward H. Shortliffe

In this chapter MYCIN's implementation is presented in considerable detail. Our goals are to explain the data and control structures used by the program and to describe some of the complex and often unexpected problems that arose during system implementation. In Chapter 1 the motivations behind many of MYCIN's capabilities were mentioned. The reader is encouraged to bear those design criteria in mind throughout this chapter.

This chapter specifically describes the *Consultation System.* This subprogram uses both system knowledge from the corpus of rules and patient data entered by the physician to generate advice for the user. Furthermore, the program maintains a dynamic data base, which provides an ongoing record of the current consultation. As a result, this chapter must discuss both the nature of the various data structures and how they are used or maintained by the Consultation System.

Section 5.1 describes the corpus of rules and the associated data structures. It provides a formal description of the rules used by MYCIN. Our quantitative truth model is briefly introduced, and the mechanism for rule evaluation is explained. This section also describes the clinical parameters with which MYCIN is familiar and which form the basis for the conditional expressions in the premise of a rule.

In Section 5.2 MYCIN's goal-oriented control structure is described. Mechanisms for rule invocation and question selection are explained at that time. The section also discusses the creation of the dynamic data base,

This chapter is condensed from Chapter 3 of *Computer-Based Medical Consultations: MYCIN.* New York: Elsevier/North-Holland, 1976.

which is the foundation for both the system's advice and its explanation capabilities (to be described in Part Six).

Section 5.3 is devoted to an explanation of the program's context tree, i.e., the network of interrelated organisms, drugs, and cultures that characterize the patient and his or her current clinical condition. The need for such a data structure is clarified, and the method for propagation (growth) of the tree is described.

The final tasks in MYCIN's clinical problem area are the identification of potentially useful drugs and the selection of the best drug or drugs from that list. MYCIN's early mechanism for making these decisions is discussed in Section 5.4 of this chapter. Later refinements are the subject of Chapter 6.

Section 5.5 discusses MYCIN's mechanisms for storing patient data and for permitting a user to change the answer to a question. As will be described, these two capabilities are closely interrelated.

In Section 5.6 we briefly mention extensions to the system that were contemplated when this material was written in 1975. Several of these capabilities were eventually implemented.

5.1 System Knowledge

5.1.1 Decision Rules

Automated problem-solving systems use criteria for drawing conclusions that often support a direct analogy to the rule-based knowledge representation used by MYCIN. Consider, for example, the conditional probabilities that underlie Bayesian diagnosis programs. Each probability provides information that may be stated in an explicit rule format:

$P(h|e) = X$ means
IF: e is known to be true
THEN: conclude that h is true with probability X

It is important to note, therefore, that the concept of rule-based knowledge is not unique, even for medical decision-making programs.

Representation of the Rules

The 200 rules in the original MYCIN system consisted of a *premise*, an *action*, and sometimes an *else* clause. Else clauses were later deleted from the system because they were seldom used, and a general representation

of inference statements could be achieved without them. Every rule has a name of the form RULE### where ### represents a three-digit number.

The details of rules and how they are used are discussed throughout the remainder of this chapter. We therefore offer a formal definition of rules, which will serve in part as a guide for what is to follow. The rules are stored as LISP data structures in accordance with the following Backus-Nauer Form (BNF) description:

<rule> ::= <premise> <action> | <premise> <action> <else>

<premise> ::= ($AND <condition> . . . <condition>)

<condition> ::= (<func1> <context> <parameter>) |
(<func2> <context> <parameter> <value>) |
(<special-func> <arguments>) |
($OR <condition> . . . <condition>)

<action> ::= <concpart>

<else> ::= <concpart>

<concpart> ::= <conclusion> | <actfunc> |
(DO-ALL <conclusion> . . . <conclusion>) |
(DO-ALL <actfunc> . . . <actfunc>)

<context> ::= *see* Section 5.1.2

<parameter> ::= *see* Section 5.1.3

<value> ::= *see* Section 5.1.4

<func1> ::= *see* Section 5.1.5

<func2> ::= *see* Section 5.1.5

<special-func> ::= *see* Section 5.1.6

<arguments> ::= *see* Section 5.1.6

<conclusion> ::= *see* Section 5.2.3

<actfunc> ::= *see* Section 5.4

Thus the premise of a rule consists of a conjunction of conditions, each of which must hold for the indicated action to be taken. Negations of conditions are handled by individual predicates (<func1> and <func2>) and therefore do not require a $NOT function to complement the Boolean functions $AND and $OR. If the premise of a rule is known to be false, the conclusion or action indicated by the else clause is taken. If the truth

of the premise cannot be ascertained or the premise is false but no else condition exists, the rule is simply ignored.

The premise of a rule is always a conjunction of one or more conditions. Disjunctions of conditions may be represented as multiple rules with identical action clauses. A condition, however, may itself be a disjunction of conditions. These conventions are somewhat arbitrary but do provide sufficient flexibility so that any Boolean expression may be represented by one or more rules. As is discussed in Section 5.2, multiple rules are effectively ORed together by MYCIN's control structure.

For example, two-leveled Boolean nestings of conditions are acceptable as follows:

Legal:

[1] A & B & C → D

[2] A & (B or C) → D

[3] (A or B or C) & (D or E) → F

Illegal:

[4] A or B or C → D

[5] A & (B or (C & D)) → E

Rule [4] is correctly represented by the following three rules:

[6] A → D

[7] B → D

[8] C → D

whereas [5] must be written as:

[9] A & C & D → E

[10] A & B → E

Unlike rules that involve strict implication, MYCIN's rules allow the strength of an inference to be modified by a certainty factor (CF). A CF is a number from −1 to +1, the nature of which is described in Section 5.1.4 and in Chapter 11.

The following three examples are rules from MYCIN that have been translated into English from their internal LISP representation (Section 5.1.7). They represent the range of rule types available to the system. The details of their internal representation will be explained as we proceed.

RULE037

IF: 1) *The identity of the organism is not known with certainty, and*
2) The stain of the organism is gramneg, and
3) The morphology of the organism is rod, and
4) The aerobicity of the organism is aerobic
THEN: There is strongly suggestive evidence (.8) that the class of the organism is enterobacteriaceae

RULE145

IF: 1) The therapy under consideration is one of: cephalothin clindamycin erythromycin lincomycin vancomycin, and
2) Meningitis is an infectious disease diagnosis *for the patient*
THEN: It is definite (1) the therapy under consideration is not a potential therapy for use against the organism

RULE060

IF: *The identity of the organism is* bacteroides
THEN: I recommend therapy chosen from among the following drugs:
1 - clindamycin (.99)
2 - chloramphenicol (.99)
3 - erythromycin (.57)
4 - tetracycline (.28)
5 - carbenicillin (.27)

Before we can explain how rules such as these are invoked and evaluated, it is necessary to describe further MYCIN's internal organization. We shall therefore temporarily digress in order to lay some groundwork for the description of the evaluation functions in Section 5.1.5.

5.1.2 Categorization of Rules by Context

The Context Tree

Although it is common to describe diagnosis as inference based on attributes of the patient, MYCIN's decisions must necessarily involve not only the patient but also the cultures that have been grown, organisms that have been isolated, and drugs that have been administered. Each of these is termed a *context* of the program's reasoning (see <context> in the BNF description of rules).[1]

MYCIN currently (1975) knows about ten different context-types:

[1]The use of the word *context* should not be confused with its meaning in high-level languages that permit temporary saving of all information regarding a program's current status—a common mechanism for backtracking and parallel-processing implementations.

CURCULS	A current culture from which organisms were isolated
CURDRUGS	An antimicrobial agent currently being administered to a patient
CURORGS	An organism isolated from a current culture
OPDRGS	An antimicrobial agent administered to the patient during a recent operative procedure
OPERS	An operative procedure the patient has undergone
PERSON	The patient
POSSTHER	A therapy being considered for recommendation
PRIORCULS	A culture obtained in the past
PRIORDRGS	An antimicrobial agent administered to the patient in the past
PRIORORGS	An organism isolated from a prior culture

Except for PERSON, each of these context-types may be instantiated more than once during any given run of the consultation program. Some may not be created at all if they do not apply to the given patient. However, each time a context-type is instantiated, it is given a unique name. For example, CULTURE-1 is the first CURCUL and ORGANISM-1 is the first CURORG. Subsequent CURCULS or PRIORCULS are called CULTURE-2, CULTURE-3, etc.

The context-types instantiated during a run of the consultation program are arranged hierarchically in a data structure termed the *context tree*. One such tree is shown in Figure 5-1. The context-type for each instantiated context is shown in parentheses near its name. Thus, to clarify terminology, we note that a node in the context tree is called a *context* and is created as an instantiation of a context-type. This sample context tree corresponds to a patient from whom two current cultures and one prior culture were obtained. One organism was isolated from each of the current cultures, but the patient is being treated (with two drugs) for only one of the current organisms. Furthermore, two organisms were grown from the prior culture, but therapy was instituted to combat only one of these. Finally, the patient has had a recent operative procedure during which he or she was treated with an antimicrobial agent.

The context tree is useful not only because it gives structure to the clinical problem (Figure 5-1 already tells us a good deal about PATIENT-1), but also because we often need to be able to relate one context to another. For example, in considering the significance of ORGANISM-2, MYCIN may well want to be able to reference the site of the culture from which ORGANISM-2 was obtained. Since the patient has had three different cultures, we need an explicit mechanism for recognizing that ORGANISM-2 came from CULTURE-2, not from CULTURE-1 or CULTURE-3. The technique for dynamic propagation (i.e., growth) of the context tree during a consultation is described in Section 5.3.

SAMPLE CONTEXT TREE

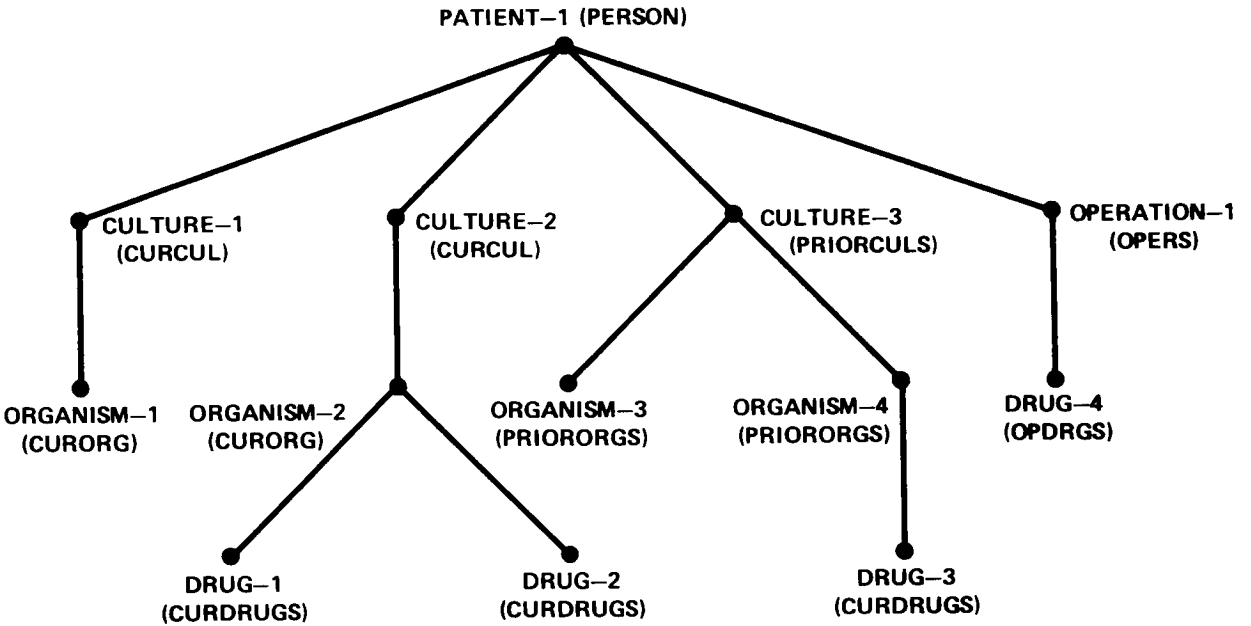

FIGURE 5-1 Context tree for a sample patient with two recent positive cultures, an older one, and a recent significant operative procedure. Nodes in the tree are termed *contexts.*

Interrelationship of Rules and the Tree

The 200 rules currently used by MYCIN[2] are not explicitly linked in a decision tree or reasoning network. This feature is in keeping with our desire to keep system knowledge modular and manipulable. However, rules are subject to categorization in accordance with the context-types for which they are most appropriately invoked. For example, some rules deal with organisms, some with cultures, and still others deal solely with the patient. MYCIN's current rule categories are as follows (context-types to which they may be applied are enclosed in parentheses):

CULRULES	Rules that may be applied to any culture (CURCULS or PRIORCULS)
CURCULRULES	Rules that may be applied only to current cultures (CURCULS)
CURORGRULES	Rules that may be applied only to current organisms (CURORGS)
DRGRULES	Rules that may be applied to any antimicrobial agent that has been administered to combat a specific organism (CURDRUGS or PRIORDRGS)
OPRULES	Rules that may be applied to operative procedures (OPERS)
ORDERRULES	Rules that are used to order the list of possible therapeutic recommendations (POSSTHER)
ORGRULES	Rules that may be applied to any organism (CURORGS or PRIORORGS)
PATRULES	Rules that may be applied to the patient (PERSON)
PDRGRULES	Rules that may be applied only to drugs given to combat prior organisms (PRIORDRGS)
PRCULRULES	Rules that may be applied only to prior cultures (PRIORCULS)
PRORGRULES	Rules that may be applied only to organism isolated from prior cultures (PRIORORGS)
THERULES	Rules that store information regarding drugs of choice (Section 5.4.1)

Every rule in the MYCIN system belongs to one, and only one, of these categories. Furthermore, selecting the proper category for a newly acquired rule does not present a problem. In fact, category selection can be automated to a large extent.

Consider a rule such as this:

---

[2]*Ed. note:* This number increased to almost 500 by 1978.

RULE124

IF: 1) The site of the culture is throat, and
2) The identity of the organism is streptococcus
THEN: There is strongly suggestive evidence (.8) that
the subtype of the organism is not group-D

This is one of MYCIN's ORGRULES and may thus be applied to either a CURORGS context or a PRIORORGS context. Referring back to Figure 5-1, suppose RULE124 were applied to ORGANISM-2. The first condition in the premise refers to the site of the culture from which ORGANISM-2 was isolated (i.e., CULTURE-2) and not to the organism itself (i.e., organisms do not have sites, but cultures do). The context tree is therefore important for determining the proper context when a rule refers to an attribute of a node in the tree other than the context to which the rule is being explicitly applied. Note that this means that a single rule may refer to nodes at several levels in the context tree. The rule is categorized simply on the basis of the lowest context-type (in the tree) that it may reference. Thus RULE124 is an ORGRULE rather than a CULRULE.

5.1.3 Clinical Parameters

This subsection describes the data types indicated by <parameter> and <value> in the BNF description of rules. Although we have previously asserted that all MYCIN's knowledge is stored in its corpus of rules, the clinical parameters and their associated properties comprise an important class of second-level knowledge. We shall first explain the kind of parameters used by the system and then describe their representation.

A clinical parameter is a characteristic of one of the contexts in the context tree, i.e., the name of the patient, the site of a culture, the morphology of an organism, the dose of a drug, etc. A patient's status would be completely specified by a context tree in which values were known for all the clinical parameters characterizing each node in the tree (assuming the parameters known to MYCIN encompass all those that are clinically relevant—a dubious assumption at present). In general, this is more information than is needed, however, so one of MYCIN's tasks is to identify those clinical parameters that need to be considered for the patient about whom advice is being sought.

The concept of an attribute-object-value triple is common within the AI field. This associative relationship is a basic data type for the SAIL language (Feldman et al., 1972) and is the foundation for the property-list formalism in LISP (McCarthy et al., 1962). Relational predicates in predicate calculus also represent associative triples. The point is that many facts may be expressed as triples that state that some object has an attribute with some specified value. Stated in the order <attribute object value>, examples include:

(COLOR BALL RED)
(OWNS FIREMAN RED-SUSPENDERS)

```
(AGE BOB 22)
(FATHER CHILD 'DADDY')
(GRAMSTAIN ORGANISM GRAM-POSITIVE)
(DOSE DRUG 1.5-GRAMS)
(MAN BOB TRUE)
(WOMAN BOB FALSE)
```

Note that the last two examples are different from the others in that they represent a rather different kind of relationship. In fact, several authors would classify the first six as "relations" and the last two as "predicates," using the simpler notation:

```
MAN (BOB)
-WOMAN (BOB)
```

Regardless of whether it is written as MAN(BOB) or (MAN BOB TRUE), this binary predicate statement has rather different characteristics from the relations that form natural triples. This distinction will become clearer later (see yes-no parameters below).

MYCIN stores inferences and data using the attribute-object-value concept. The object is always some context in the context tree, and the attribute is a clinical parameter appropriate for that context. Information stored using this mechanism may be retrieved and updated in accordance with a variety of conventions described throughout this chapter.

The Three Kinds of Clinical Parameters

There are three fundamentally different kinds of clinical parameters. The simplest variety is *single-valued parameters*. These are attributes such as the name of the patient and the identity of the organism. In general, they have a large number of possible values that are mutually exclusive. As a result, only one can be the true value, although several may seem likely at any point during the consultation.

Multi-valued parameters also generally have a large number of possible values. The difference is that the possible values need not be mutually exclusive. Thus such attributes as a patient's drug allergies and a locus of an infection may have multiple values, each of which is known to be correct.

The third kind of clinical parameter corresponds to the binary predicate discussed above. These are attributes that are either true or false for the given context. For example, the significance of an organism is either true or false (yes or no), as is the parameter indicating whether the dose of a drug is adequate. Attributes of this variety are called *yes-no parameters*. They are, in effect, a special kind of single-valued parameter for which there are only two possible values.

Classification and Representation of the Parameters

The clinical parameters known to MYCIN are categorized in accordance with the context to which they apply. These categories include:

PROP-CUL	Those clinical parameters which are attributes of cultures (e.g., site of the culture, method of collection)
PROP-DRG	Those clinical parameters which are attributes of administered drugs (e.g., name of the drug, duration of administration)
PROP-OP	Those clinical parameters which are attributes of operative procedures (e.g., the cavity, if any, opened during the procedure)
PROP-ORG	Those clinical parameters which are attributes of organisms (e.g., identity, gram stain, morphology)
PROP-PT	Those clinical parameters which are attributes of the patient (e.g., name, sex, age, allergies, diagnoses)
PROP-THER	Those clinical parameters which are attributes of therapies being considered for recommendation (e.g., recommended dosage, prescribing name)

These categories encompass all clinical parameters used by the system. Note that any of the nodes (contexts) in the context tree for the patient may be fully characterized by the values of the set of clinical parameters in one of these categories.

Each of the 65 clinical parameters currently (1975) known to MYCIN has an associated set of properties that is used during consideration of the parameter for a given context. Figure 5-2 presents examples of the three types of clinical parameters, which together demonstrate several of these properties:

EXPECT	This property indicates the range of expected values that the parameter may have. IF equal to (YN), then the parameter is a yes-no parameter. IF equal to (NUMB), then the expected value of the parameter is a number. IF equal to (ONE-OF <list>), then the value of the parameter must be a member of <list>. IF equal to (ANY), then there is no restriction on the range of values that the parameter may have.
PROMPT	This property is a sentence used by MYCIN when it requests the value of the clinical parameter from the user; if there is an asterisk in the phrase (see Figure 5-2), it is replaced by the name of the context about which the question is being asked; this property is used only for yes-no or single-valued parameters.
PROMPT1	This property is similar to PROMPT but is used if the clinical parameter is a multi-valued parameter; in these cases MYCIN only asks the question about

Yes-No Parameter

FEBRILE: <FEBRILE is an attribute of a patient and is therefore a member of the list PROP-PT>

EXPECT: (YN)
LOOKAHEAD: (RULE149 RULE109 RULE045)
PROMPT: (Is * febrile?)
TRANS: (* IS FEBRILE)

Single-Valued Parameter

IDENT: <IDENT is an attribute of an organism and is therefore a member of the list PROP-ORG>

CONTAINED-IN: (RULE030)
EXPECT: (ONEOF (ORGANISMS))
LABDATA: T
LOOKAHEAD: (RULE004 RULE054 . . . RULE168)
PROMPT: (Enter the identity (genus) of *:)
TRANS: (THE IDENTITY OF *)
UPDATED-BY: (RULE021 RULE003 . . . RULE166)

Multi-Valued Parameter

INFECT: <INFECT is an attribute of a patient and is therefore a member of the list PROP-PT>

EXPECT: (ONEOF (PERITONITIS BRAIN-ABCESS MENINGITIS BACTEREMIA UPPER-URINARY-TRACT-INFECTION . . . ENDOCARDITIS))
LOOKAHEAD: (RULE115 RULE149 . . . RULE045)
PROMPT1: (Is there evidence that the patient has a (VALU)?)
TRANS: (AN INFECTIOUS DISEASE DIAGNOSIS FOR *)
UPDATED-BY: (RULE157 RULE022 . . . RULE105)

FIGURE 5-2 Examples of the three types of clinical parameters. As shown, each clinical parameter is characterized by a set of properties described in the text.

	a single one of the possible parameter values; the value of interest is substituted for (VALU) in the question.
LABDATA	This property is a flag, which is either T or NIL; if T it indicates that the clinical parameter is a piece of primitive data, the value of which may be known with certainty to the user (see Section 5.2.2).
LOOKAHEAD	This property is a list of all rules in the system that reference the clinical parameter in the premise.

UPDATED-BY	This property is a list of all rules in the system in which the action or else clause permits a conclusion to be made regarding the value of the clinical parameter.
CONTAINED-IN	This property is a list of all rules in the system in which the action or else clause references the clinical parameter but does not cause its value to be updated.
TRANS	This property is used to translate an occurrence of this parameter into its English representation; the context of the parameter is substituted for the asterisk during translation.
DEFAULT	This property is used only with clinical parameters for which EXPECT = (NUMB); it gives the expected units for numerical answers (days, years, grams, etc.).
CONDITION	This property, when utilized, is an executable LISP expression that is evaluated before MYCIN requests the value of the parameter; if the CONDITION is true, the question is not asked (e.g., "Don't ask for an organism's subtype if its genus is not known by the user").

The uses of these properties will be discussed throughout the remainder of this chapter. However, a few additional points are relevant here. First, it should be noted that the order of rules for the properties LOOK-AHEAD, UPDATED-IN, and CONTAINED-IN is arbitrary and does not affect the program's advice. Second, EXPECT and TRANS are the only properties that *must* exist for every clinical parameter. Thus, for example, if there is no PROMPT or PROMPT1 stored for a parameter, the system assumes that it simply cannot ask the user for the value of the parameter. Finally, note in Figure 5-2 the difference in the TRANS property for yes-no and non–yes-no parameters. In general, a parameter and its value may be translated as follows:

THE <attribute> OF <object> IS <value>

However, for a yes-no parameter such as FEBRILE, it is clearly necessary to translate the parameter in a fashion other than this:

THE FEBRILE OF PATIENT-1 IS YES

Our solution has been to suppress the YES altogether and simply to say:

PATIENT-1 IS FEBRILE

5.1.4 Certainty Factors

Chapter 11 presents a detailed description of certainty factors and their theoretical foundation. This section therefore provides only a brief overview of the subject. A familiarity with the characteristics of certainty factors (CF's) is necessary for the discussion of MYCIN during the remainder of this chapter.

The value of every clinical parameter is stored by MYCIN along with an associated certainty factor that reflects the system's "belief" that the value is correct. This formalism is necessary because, unlike domains in which objects either have or do not have some attribute, in medical diagnosis and treatment there is often uncertainty regarding attributes such as the significance of the disease, the efficacy of a treatment, or the diagnosis itself. CF's are an alternative to conditional probability that has several advantages in MYCIN's domain.

A certainty factor is a number between -1 and $+1$ that reflects the degree of belief in a hypothesis. Positive CF's indicate there is evidence that the hypothesis is valid. The larger the CF, the greater is the belief in the hypothesis. When CF $= 1$, the hypothesis is known to be correct. On the other hand, negative CF's indicate that the weight of evidence suggests that the hypothesis is false. The smaller the CF, the greater is the belief that the hypothesis is invalid. CF $= -1$ means that the hypothesis has been effectively disproven. When CF $= 0$, there is either no evidence regarding the hypothesis or the supporting evidence is equally balanced by evidence suggesting that the hypothesis is not true.

MYCIN's hypotheses are statements regarding values of clinical parameters for the various nodes in the context tree. For example, sample hypotheses are

h_1 = The identity of ORGANISM-1 is streptococcus
h_2 = PATIENT-1 is febrile
h_3 = The name of PATIENT-1 is John Jones

We use the notation CF[h,E]=X to represent the certainty factor for the hypothesis h based on evidence E. Thus, if CF[h_1,E] $= .8$, CF[h_2,E] $= -.3$, and CF[h_3,E] $= +1$, the three sample hypotheses above may be qualified as follows:

CF[h_1,E] $= .8$: There is strongly suggestive evidence (.8) that the identity of ORGANISM-1 is streptococcus

CF[h_2,E] $= -.3$: There is weakly suggestive evidence (.3) that PATIENT-1 is not febrile

CF[h_3,E] $= +1$: It is definite (1) that the name of PATIENT-1 is John Jones

Certainty factors are used in two ways. First, as noted, the value of every clinical parameter is stored with its associated certainty factor. In this case the evidence E stands for all information currently available to MY-

CIN. Thus, if the program needs the identity of ORGANISM-1, it may look in its dynamic data base and find:

IDENT of ORGANISM-1 = ((STREPTOCOCCUS .8))

The second use of CF's is in the statement of decision rules themselves. In this case the evidence E corresponds to the conditions in the premise of the rule. Thus

$$A \ \& \ B \ \& \ C \xrightarrow{X} D$$

is a representation of the statement CF[D,(A & B & C)] = X. For example, consider the following rule:

IF: 1) The stain of the organism is grampos, and
2) The morphology of the organism is coccus, and
3) The growth conformation of the organism is chains
THEN: There is suggestive evidence (.7) that the identity of the organism is streptococcus

This rule may also be represented as $CF[h_1,e] = .7$, where h_1 is the hypothesis that the organism (context of the rule) is a *Streptococcus* and e is the evidence that it is a gram-positive coccus growing in chains.

Since diagnosis is, in effect, the problem of selecting a disease from a list of competing hypotheses, it should be clear that MYCIN may simultaneously be considering several hypotheses regarding the value of a clinical parameter. These hypotheses are stored together, along with their CF's, for each node in the context tree. We use the notation Val[C,P] to signify the set of all hypotheses regarding the value of the clinical parameter P for the context C. Thus, if MYCIN has reason to believe that ORGANISM-1 may be either a *Streptococcus* or a *Staphylococcus,* but *Pneumococcus* has been ruled out, its dynamic data base might well show:

Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS .6)(STAPHYLOCOCCUS .4)
(DIPLOCOCCUS-PNEUMONIAE -1))

It can be shown that the sum of the CF's for supported hypotheses regarding a single-valued parameter (i.e., those parameters for which the hypotheses are mutually exclusive) cannot exceed 1 (Shortliffe and Buchanan, 1975). Multi-valued parameters, on the other hand, may have several hypotheses that are all known to be true, for example:

Val[PATIENT-1,ALLERGY] = ((PENICILLIN 1)(AMPICILLIN 1)
(CARBENICILLIN 1)(METHICILLIN 1))

As soon as a hypothesis regarding a single-valued parameter is proved to be true, all competing hypotheses are effectively disproved:

Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS 1)(STAPHYLOCOCCUS -1)
(DIPLOCOCCUS-PNEUMONIAE -1))

In Chapter 11 we demonstrate that CF[h,E] = −CF[¬h,E]. This observation has important implications for the way MYCIN handles the binary-valued attributes we call yes-no parameters. Since "yes" is "¬no," it is not necessary to consider "yes" and "no" as competing hypotheses for the value of a yes-no parameter (as we do for single-valued parameters). Instead, we can always express "no" as "yes" with a reversal in the sign of the CF. This means that Val[C,P] is always equal to the single value "yes," along with its associated CF, when P is a yes-no parameter.

We discuss below MYCIN's mechanism for adding to the list of hypotheses in Val[C,P] as new rules are invoked and executed. However, the following points should be emphasized here:

1. The strength of the conclusion associated with the execution of a rule reflects not only the CF assigned to the rule, but also the program's degree of belief regarding the validity of the premise.
2. The support of several rules favoring a single hypothesis may be assimilated incrementally on the list Val[C,P] by using the special combining functions described in Chapter 11.

5.1.5 Functions for the Evaluation of Premise Conditions

This section describes the evaluation of the individual conditions (see <condition>, Section 5.1.1) in the premise of rules. Conditions in general evaluate to true or false (T or NIL). Thus they may at first glance be considered simple predicates on the values of clinical parameters. However, since there may be several competing hypotheses on the list Val[C,P], each associated with its own degree of belief as reflected by the CF, conditional statements regarding the value of parameters can be quite complex. All predicates are implemented as LISP functions. The functions that undertake the required analysis are of three varieties, specified by the designations <func1>, <func2>, and <special-func> in the BNF rule description. This section explains the <func1> and <func2> predicates. The <special-func> category is deferred until later, however, so that we may first introduce our specialized knowledge structures.

There are four predicates in the category <func1>. These functions do not form conditionals on specific values of a clinical parameter but are concerned with the more general status of knowledge regarding the attributes in question. For example, KNOWN[ORGANISM-1,IDENT] is an invocation of the <func1> predicate KNOWN; it would return true if the identity of ORGANISM-1 were known, regardless of the value of the clinical parameter IDENT. KNOWN and the other <func1> predicates may be formally defined as follows:

Predicates of the Category <func1>

Let V = Val[C,P] be the set of all hypotheses regarding the value of the clinical parameter P for the context C.

Let Mv = Max[V] be the most strongly supported hypothesis in V (i.e., the hypothesis with the largest CF).

Let CFmv = CF[Mv,E] where E is the total available evidence.

Then, if P is either a single-valued or multi-valued parameter, the four predicates (functions) may be specified as follows:

Function	*If*	*Then*	*Else*
KNOWN[C,P]	CFmv > .2	T	NIL
NOTKNOWN[C,P]	CFmv ≤ .2	T	NIL
DEFINITE[C,P]	CFmv = 1	T	NIL
NOTDEFINITE[C,P]	CFmv < 1	T	NIL

In words, these definitions reflect MYCIN's convention that the value of a parameter is known if the CF of the most highly supported hypothesis exceeds .2. The .2 threshold was selected empirically. The implication is that a positive CF less than .2 reflects so little evidence supporting the hypothesis that there is virtually no reasonable hypothesis currently known. The interrelationships among these functions are diagrammed on a CF number line in Figure 5-3. Regions specified are the range of values for CFmv over which the function returns T.

As was pointed out in the preceding section, however, yes-no parameters are special cases because we know CF[YES,E] = −CF[NO,E]. Since the values of yes-no parameters are always stored in terms of YES, MYCIN must recognize that a YES with CF = −.9 is equivalent to a NO with CF = .9. The definitions of the four <func1> predicates above do not reflect this distinction. Therefore, when P is a yes-no parameter, the four functions are specified as follows:

Function	*If*	*Then*	*Else*
KNOWN[C,P]	|CFmv| > .2	T	NIL
NOTKNOWN[C,P]	|CFmv| ≤ .2	T	NIL
DEFINITE[C,P]	|CFmv| = 1	T	NIL
NOTDEFINITE[C,P]	|CFmv| < 1	T	NIL

Figure 5-4 shows the relationship among these functions for yes-no parameters.

There are nine predicates in the category <func2>. Unlike the <func1> predicates, these functions control conditional statements regarding specific values of the clinical parameter in question. For example, SAME[ORGANISM-1,IDENT,E.COLI] is an invocation of the <func2>

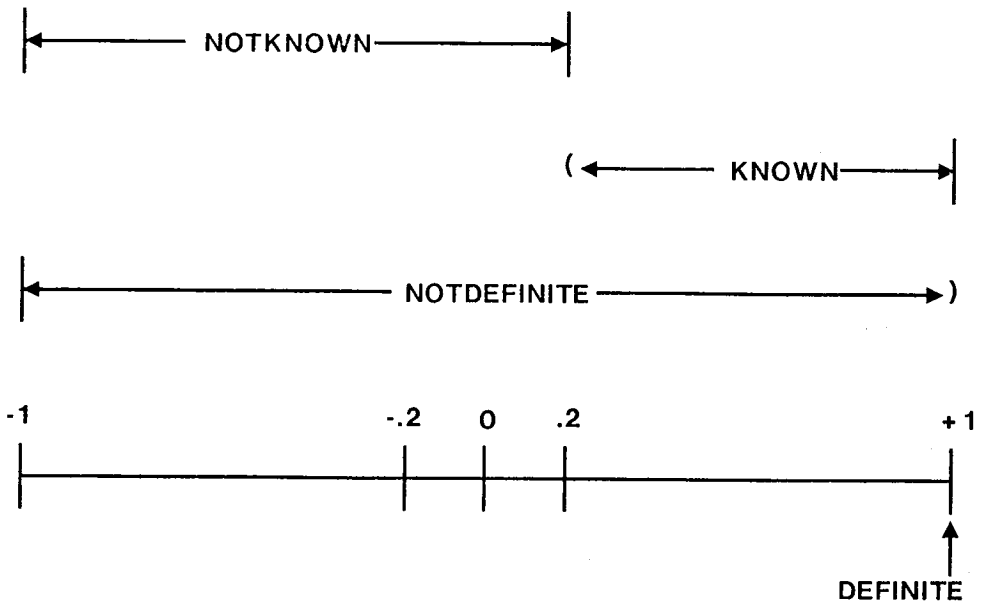

FIGURE 5-3 Diagram indicating the range of CF values over which the <func1> predicates hold true when applied to multi-valued or single-valued (i.e., non–yes-no) clinical parameters. Vertical lines and parentheses distinguish closed and non-closed certainty factor ranges, respectively.

predicate SAME; it would return a non-NIL value if the identity of ORGANISM-1 were known to be *E. coli*. SAME and the other <func2> predicates may be formally defined as follows:

Predicates of the Category <func2>

Let V = Val[C,P] be the set of all hypotheses regarding the value of the clinical parameter P for the context C.

Let I = Intersection[V,LST] be the set of all hypotheses in V that also occur in the set LST; LST contains the possible values of P for comparison by the predicate function; it usually contains only a single element; if no element in LST is also in V, I is simply the empty set.

Let M_i = Max[I] be the most strongly confirmed hypothesis in I; thus M_i is NIL if I is the empty set.

Let CFmi = CF[M_i,E] where CFmi = 0 if M_i is NIL.

Then the <func2> predicates are specified as follows:

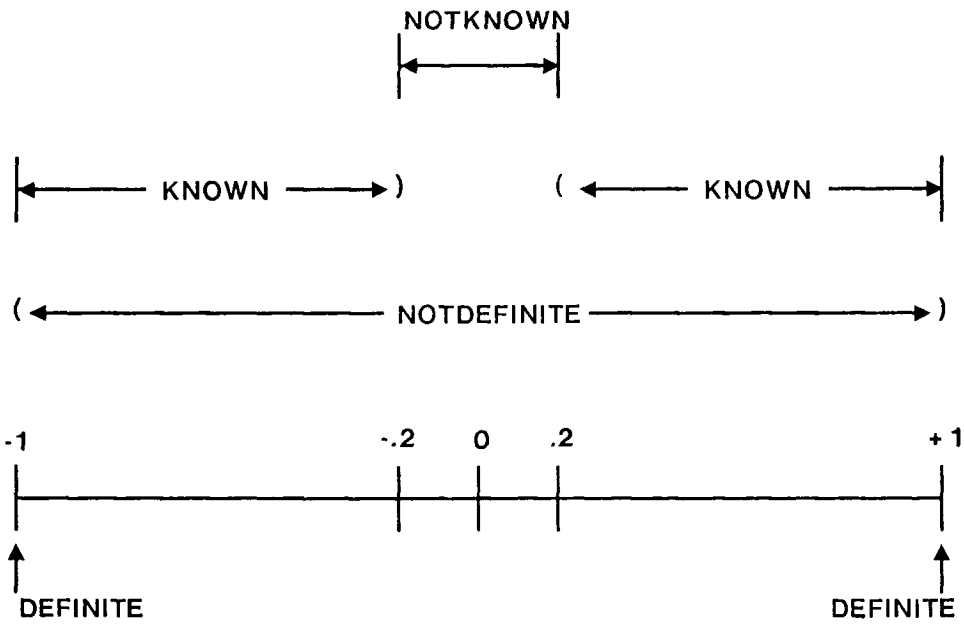

FIGURE 5-4 Diagram indicating the range of CF values over which the <func1> predicates hold true when applied to yes-no clinical parameters.

Function	*If*	*Then*	*Else*
SAME[C,P,LST]	CFmi > .2	CFmi	NIL
THOUGHTNOT[C,P,LST]	CFmi < −.2	−CFmi	NIL
NOTSAME[C,P,LST]	CFmi ≤ .2	T	NIL
MIGHTBE[C,P,LST]	CFmi ≥ −.2	T	NIL
VNOTKNOWN[C,P,LST]	|CFmi| ≤ .2	T	NIL
DEFIS[C,P,LST]	CFmi = +1	T	NIL
DEFNOT[C,P,LST]	CFmi = −1	T	NIL
NOTDEFIS[C,P,LST]	.2 < CFmi < 1	T	NIL
NOTDEFNOT[C,P,LST]	−1 < CFmi < −.2	T	NIL

The names of the functions have been selected to reflect their semantics. Figure 5-5 shows a graphic representation of each function and also explicitly states the interrelationships among them.

Note that SAME and THOUGHTNOT are different from all the other functions in that they return a number (CF) rather than T if the defining condition holds. This feature permits MYCIN to record the degree to which premise conditions are satisfied. In order to explain this

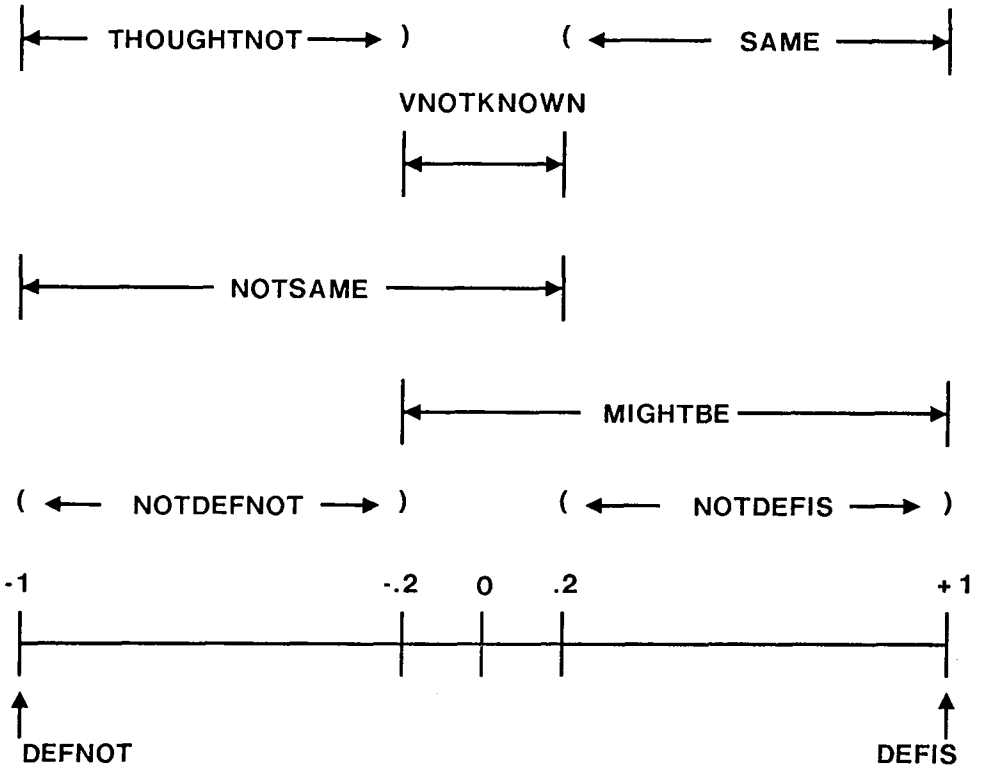

FIGURE 5-5 Diagram indicating the range of CF values over which the <func2> predicates hold true. The logical relationships of these predicates are summarized below the diagram.

point, we must discuss the $AND function that oversees the evaluation of the premise of a rule. The reader will recall the BNF description:

<premise> ::= ($AND <condition> . . . <condition>)

$AND is similar to the standard LISP AND function in that it evaluates its conditional arguments one at a time, returning false (NIL) as soon as a condition is found to be false, and otherwise returning true (T). The difference is that $AND expects some of its conditions to return numerical values rather than simply T or NIL. If an argument condition returns NIL

(or a number equal to .2 or less), it is considered false and $AND stops considering subsequent arguments. On the other hand, nonnumeric values of conditions are interpreted as indicating truth with CF = 1. Thus each true condition either returns a number or a non-NIL value that is interpreted as 1. $AND then maintains a record of the lowest value returned by any of its arguments. This number, termed TALLY, is a certainty tally, which indicates MYCIN's degree of belief in the premise (see Combining Function 2 in Chapter 11). Thus $.2 < \text{TALLY} \leq 1$, where TALLY = 1 indicates that MYCIN believes the premise to be true with certainty.

Most of the predicates that evaluate conditions in the premise of a rule return either T or NIL as we have shown. Consider, however, the semantics of the most commonly used function, SAME, and its analogous function, THOUGHTNOT. Suppose MYCIN knows:

```
Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS .7)(STAPHYLOCOCCUS .3))
```

Then it seems clear that

```
SAME[ORGANISM-1,IDENT,STREPTOCOCCUS]
```

is in some sense "more true" than

```
SAME[ORGANISM-1,IDENT,STAPHYLOCOCCUS]
```

even though both hypotheses exceed the threshold CF = .2. If SAME merely returned T, this distinction would be lost. Thus, for this example:

```
                SAME[ORGANISM-1,IDENT,STREPTOCOCCUS] = .7
                SAME[ORGANISM-1,IDENT,STAPHYLOCOCCUS] = .3
whereas         KNOWN[ORGANISM-1,IDENT] = T
and             NOTDEFIS[ORGANISM-1,IDENT,STREPTOCOCCUS] = T
```

A similar argument explains why THOUGHTNOT returns a CF rather than T. It is unclear whether any of the other <func2> predicates should return a CF rather than T; our present conviction is that the semantics of those functions do not require relative weightings in the way that SAME and THOUGHTNOT do.

Consider a brief example, then, of the way in which the premise of a rule is evaluated by $AND. The following ORGRULE:

```
IF:  1) The stain of the organism is gramneg, and
     2) The morphology of the organism is rod, and
     3) The aerobicity of the organism is aerobic
THEN: There is strongly suggestive evidence (.8) that
      the class of the organism is enterobacteriaceae
```

is internally coded in LISP as:

```
PREMISE:  ($AND (SAME CNTXT GRAM GRAMNEG)
                (SAME CNTXT MORPH ROD)
                (SAME CNTXT AIR AEROBIC))
ACTION:   (CONCLUDE CNTXT CLASS ENTEROBACTERIACEAE TALLY .8)
```

Suppose this rule has been invoked for consideration of ORGANISM-1; i.e., the context of the rule (CNTXT) is the node in the context tree termed ORGANISM-1. Now suppose that MYCIN has the following information in its data base (we will discuss later how it gets there):

```
Val[ORGANISM-1,GRAM] = ((GRAMNEG 1.0))
Val[ORGANISM-1,MORPH] = ((ROD .8)(COCCUS .2))
Val[ORGANISM-1,AIR] = ((AEROBIC .6)(FACUL .4))
```

$AND begins by evaluating SAME[ORGANISM-1,GRAM,GRAMNEG]. The function returns CF = 1.0, so TALLY is set to 1.0 (see definition of TALLY in the description of $AND above). Next $AND evaluates the second premise condition, SAME[ORGANISM-1,MORPH,ROD], which returns .8. Since the first two conditions both were found to hold, $AND evaluates SAME[ORGANISM-1,AIR,AEROBIC], which returns .6. Thus TALLY is set to .6, and $AND returns T. Since the premise is true, MYCIN may now draw the conclusion indicated in the action portion of the rule. Note, however, that CONCLUDE has as arguments both .8 (i.e., the CF for the rule as provided by the expert) and TALLY (i.e., the certainty tally for the premise). CONCLUDE and the other functions that control inferences are described later.

5.1.6 Static Knowledge Structures

Although all MYCIN's inferential knowledge is stored in rules, there are various kinds of static definitional information, which are stored differently even though they are accessible from rules.

Tabular and List-Based Knowledge

There are three categories of knowledge structures that could be discussed in this section. However, one of them, MYCIN's dictionary, is used principally for natural language understanding and will therefore not be described. The other two data structures are simple lists and knowledge tables.

Simple lists: Simple lists provide a mechanism for simplifying references to variables and optimizing knowledge storage by avoiding unnecessary duplication. Two examples should be sufficient to explain this point.

As was shown earlier, the EXPECT property for the clinical parameter IDENT is

```
(ONEOF (ORGANISMS))
```

ORGANISMS is the name of a linear list containing the names of all bac-

teria known to MYCIN. There is also a clinical parameter named COVERFOR for which the EXPECT property is

(ONEOF ENTEROBACTERIACEAE (ORGANISMS) G+COCCI C-COCCI)

Thus, by storing the organisms separately on a list named ORGANISMS, we avoid having to duplicate the list of names in the EXPECT property of both IDENT and COVERFOR. Furthermore, using the variable name rather than internal pointers to the list structure facilitates references to the list of organisms whenever it is needed.

A second example involves the several rules in the system that make conclusions based on whether an organism was isolated from a site that is normally sterile or nonsterile. STERILESITES is the name of a simple list containing the names of all normally sterile sites known to the system. There is a similar list named NONSTERILESITES. Thus many rules can have the condition (SAME CNTXT SITE STERILESITES), and the sites need not be listed explicitly in each rule.

Knowledge tables: In conjunction with the special functions discussed in the next subsection, MYCIN's knowledge tables permit a single rule to accomplish a task that would otherwise require several rules. A knowledge table contains a comprehensive record of certain clinical parameters plus the values they take on under various circumstances. For example, one of MYCIN's knowledge tables itemizes the gram stain, morphology, and aerobicity for every bacterial genus known to the system. Consider, then, the task of inferring an organism's gram stain, morphology, and aerobicity if its identity is known with certainty. Without the knowledge table, MYCIN would require several rules of the following form:

IF: The identity of the organism is definitely W
THEN: 1) It is definite (1) that the gramstain of the organism is X, and
2) It is definite (1) that the morphology of the organism is Y, and
3) It is definite (1) that the aerobicity of the organism is Z

Instead, MYCIN contains a single rule of the following form:

RULE030

IF: The identity of the organism is known with certainty
THEN: It is definite (1) that these parameters - GRAM MORPH AIR - should be transferred from the identity of the organism to this organism

Thus if ORGANISM-1 is known to be a *Streptococcus*, MYCIN can use RULE030 to access the knowledge table to look up the organism's gram stain, morphology, and aerobicity.

Specialized Functions

The efficient use of knowledge tables requires the existence of four specialized functions (the category <special-func> from Section 5.1.1). As explained below, each function attempts to add members to a list named GRIDVAL and returns T if at least one element has been found to be placed in GRIDVAL.

Functions of the Category <special-func>

Let V = Val[C,P] be the set of all hypotheses regarding the value of the clinical parameter P for the context C.

Let CLST be a list of objects that may be characterized by clinical parameters.

Let PLST be a list of clinical parameters.

Then:

Function	*Value of GRIDVAL*
SAME2[C,CLST,PLST]	{X | X ∈ CLST & (for all P in PLST) SAME [C,P,Val[X,P]]}
NOTSAME2[C,CLST,PLST]	{X | X ∈ CLST & (for at least one P in PLST) NOTSAME[C,P,Val[X,P]]}
SAME3[C,P,CLST,P*]	{X | X ∈ CLST & SAME[C,P,Val[X,P*]]}
NOTSAME3[C,P,CLST,P*]	{X | X ∈ CLST & NOTSAME [C,P,Val[X,P*]]}
GRID[<object>,<attribute>]	{X | X is a value of the <attribute> of <object>}

GRID is merely a function for looking up information in the specialized knowledge table.

The use of these functions is best explained by example. Consider the following verbalization of a rule given us by one of our collaborating experts:

> If you know the portal of entry of the current organism and also know the pathogenic bacteria normally associated with that site, you have evidence that the current organism is one of those pathogens so long as there is no disagreement on the basis of gram stain, morphology, or aerobicity.

This horrendous sounding rule is coded quite easily using SAME2[C,CLST,PLST], where C is the current organism, CLST is the list

of pathogenic bacteria normally associated with the portal of entry of C, and PLST is the set of properties (GRAM MORPH AIR). GRID is used to set up CLST. The LISP version of the rule is

```
PREMISE:  ($AND (GRID (VAL CNTXT PORTAL) PATH-FLORA)
                (SAME2 CNTXT GRIDVAL (QUOTE (GRAM MORPH AIR))))
ACTION:   (CONCLIST CNTXT IDENT GRIDVAL .8)
```

Note that GRID sets up the initial value of GRIDVAL for use by SAME2, which then redefines GRIDVAL for use in the action clause. This rule is translated (to somewhat stilted English) as follows:

```
IF:   1) The list of likely pathogens associated with the
         portal of entry of the organism is known, and
      2) This current organism and the members you are
         considering agree with respect to the following
         properties: GRAM MORPH AIR
THEN: There is strongly suggestive evidence (.8) that
      each of them is the identity of this current
      organism
```

SAME2 and NOTSAME2 can also be used for comparing the values of the same clinical parameters for two or more different contexts in the context tree, for example:

```
SAME2[ORGANISM-1 (ORGANISM-2 ORGANISM-3) (GRAM MORPH)]
```

On the other hand, SAME3 and NOTSAME3 are useful for comparing different parameters of two or more contexts. Suppose you need a predicate that returns T if the site of a prior organism (ORGANISM-2) is the same as the portal of entry of the current organism (ORGANISM-1). This is accomplished by the following:

```
SAME3[ORGANISM-1 PORTAL (ORGANISM-2) SITE]
```

5.1.7 Translation of Rules into English

Rules are translated into a subset of English using a set of recursive functions that piece together bits of text. We shall demonstrate the process using the premise condition (GRID (VAL CNTXT PORTAL) PATH-FLORA), which is taken from the rule in the preceding section.

The reader will recall that every clinical parameter has a property named TRANS that is used for translation (Section 5.1.3). In addition, every function, simple list, or knowledge table that is used by MYCIN's rules also has a TRANS property. For our example the following TRANS properties are relevant:

```
GRID:        (THE (2) ASSOCIATED WITH (1) IS KNOWN)
VAL:         (((2 1)))
PORTAL:      (THE PORTAL OF ENTRY OF *)
PATH-FLORA:  (LIST OF LIKELY PATHOGENS)
```

The numbers in the translations of functions indicate where the translation of the corresponding argument should be inserted. Thus the translation of GRID's second argument is inserted for the (2) in GRID's TRANS property. The extra parentheses in the TRANS for VAL indicate that the translation of VAL's first argument should be substituted for the asterisk in the translation of VAL's second argument. Since PORTAL is a PROP-ORG, CNTXT translates as "the organism," and the translation of (VAL CNTXT PORTAL) becomes

The portal of entry of the organism

Substituting VAL's translation for the (1) in GRID's TRANS and PATHFLORA's translation for the (2) yields the final translation of the conditional clause:

The list of likely pathogens associated with the portal of entry of the organism is known

Similarly, (GRID (VAL CNTXT CLASS) CLASSMEMBERS)

translates as: The list of members associated with the class of the organism is known

All other portions of rules use essentially this same procedure for translation. An additional complexity arises, however, if it is necessary to negate the verbs in action or else clauses when the associated CF is negative. The translator program must therefore recognize verbs and know how to negate them when evidence in a premise supports the negation of the hypothesis that is referenced in the action of the rule.

5.2 Use of the Rules to Give Advice

The discussion in Section 5.1 was limited to the various data structures used to represent MYCIN's knowledge. The present section proceeds to an explanation of how MYCIN uses that knowledge in order to give advice.

5.2.1 MYCIN's Control Structure

MYCIN's rules are directly analogous to the consequent theorems introduced by Hewitt in his PLANNER system (Hewitt, 1972). They permit a reasoning chain to grow dynamically on the basis of the user's answers to questions regarding the patient. This subsection describes that reasoning network, explaining how it grows and how MYCIN manages to ask questions only when there is a reason for doing so.

Consequent Rules and Recursion

MYCIN's task involves a four-stage decision problem:

1. Decide which organisms, if any, are causing significant disease.
2. Determine the likely identity of the significant organisms.
3. Decide which drugs are potentially useful.
4. Select the best drug or drugs.

Steps 1 and 2 are closely interrelated since determination of an organism's significance may well depend on its presumed identity. Furthermore, MYCIN must consider the possibility that the patient has an infection with an organism not specifically mentioned by the user (e.g., an occult abscess suggested by historical information or subtle physical findings). Finally, if MYCIN decides that there is no significant infection requiring antimicrobial therapy, it should skip Steps 3 and 4, advising the user that no treatment is thought to be necessary. MYCIN's task area therefore can be defined by the following rule:

RULE092

IF: 1) There is an organism which requires therapy, and
2) Consideration has been given to the possible existence of additional organisms requiring therapy, even though they have not actually been recovered from any current cultures
THEN: Do the following:
1) Compile the list of possible therapies which, based upon sensitivity data, may be effective against the organisms requiring treatment, and
2) Determine the best therapy recommendations from the compiled list
OTHERWISE: Indicate that the patient does not require therapy

This rule is one of MYCIN's PATRULES (i.e., its context is the patient) and is known as the *goal rule* for the system. A consultation session with MYCIN results from a simple two-step procedure:

1. Create the patient context as the top node in the context tree (see Section 5.3 for an explanation of how nodes are added to the tree).
2. Attempt to apply the goal rule to the newly created patient context.

After the second step, the consultation is over. Thus we must explain how the simple attempt to apply the goal rule to the patient causes a lengthy consultation with an individualized reasoning chain.

When MYCIN first tries to evaluate the premise of the goal rule, the first condition requires that it know whether there is an organism that requires therapy. MYCIN then reasons backwards in a manner that may be informally paraphrased as follows:

> How do I decide whether there is an organism requiring therapy? Well, RULE090 tells me that organisms associated with significant disease require therapy. But I don't even have any organisms in the context tree yet, so I'd better ask first if there are any organisms, and if there are I'll try to apply RULE090 to each of them. However, the premise of RULE090 requires that I know whether the organism is significant. I have a bunch of rules for making this decision (RULE038 RULE042 RULE044 RULE108 RULE122). For example, RULE038 tells me that if the organism came from a sterile site it is probably significant. Unfortunately, I don't have any rules for inferring the site of a culture, however, so I guess I'll have to ask the user for this information when I need it . . .

This goal-oriented approach to rule invocation and question selection is automated via two interrelated procedures, a MONITOR that analyzes rules and a FINDOUT mechanism that searches for data needed by the MONITOR.

The MONITOR analyzes the premise of a rule, condition by condition, as shown in Figure 5-6.[3] When the value of the clinical parameter referenced in a condition is not yet known to MYCIN, the FINDOUT mechanism is invoked in an attempt to obtain the missing information. FINDOUT then either derives the necessary information (from other rules) or asks the user for the data.

FINDOUT has a dual strategy depending on the kind of information required by the MONITOR. This distinction is demonstrated in Figure 5-7. In general, a piece of data is immediately requested from the user (an ASK1 question) if it is considered in some sense "primitive," as are, for example, most laboratory data. Thus, if the physician knows the identity of an organism (e.g., from a lab report), we would prefer that the system request that information directly rather than try to deduce it via decision rules. However, if the user does not know the identity of the organism, MYCIN uses its knowledge base in an effort to deduce the range of likely organisms. Nonlaboratory data are those kinds of information that require inference even by the clinician, e.g., whether or not an organism is a contaminant or whether or not a previously administered drug was effective. FINDOUT always attempts to deduce such information first, asking the physician only when MYCIN's knowledge base of rules is inadequate for making the inference from the information at hand (an ASK2 question).

We have previously described the representation of clinical parameters and their associated properties. The need for two of these properties, LABDATA and UPDATED-BY, should now be clear. The LABDATA flag for a parameter allows FINDOUT to decide which branch to take through

[3]As discussed in Section 5.1.5, the MONITOR uses the $AND function to oversee the premise evaluation.

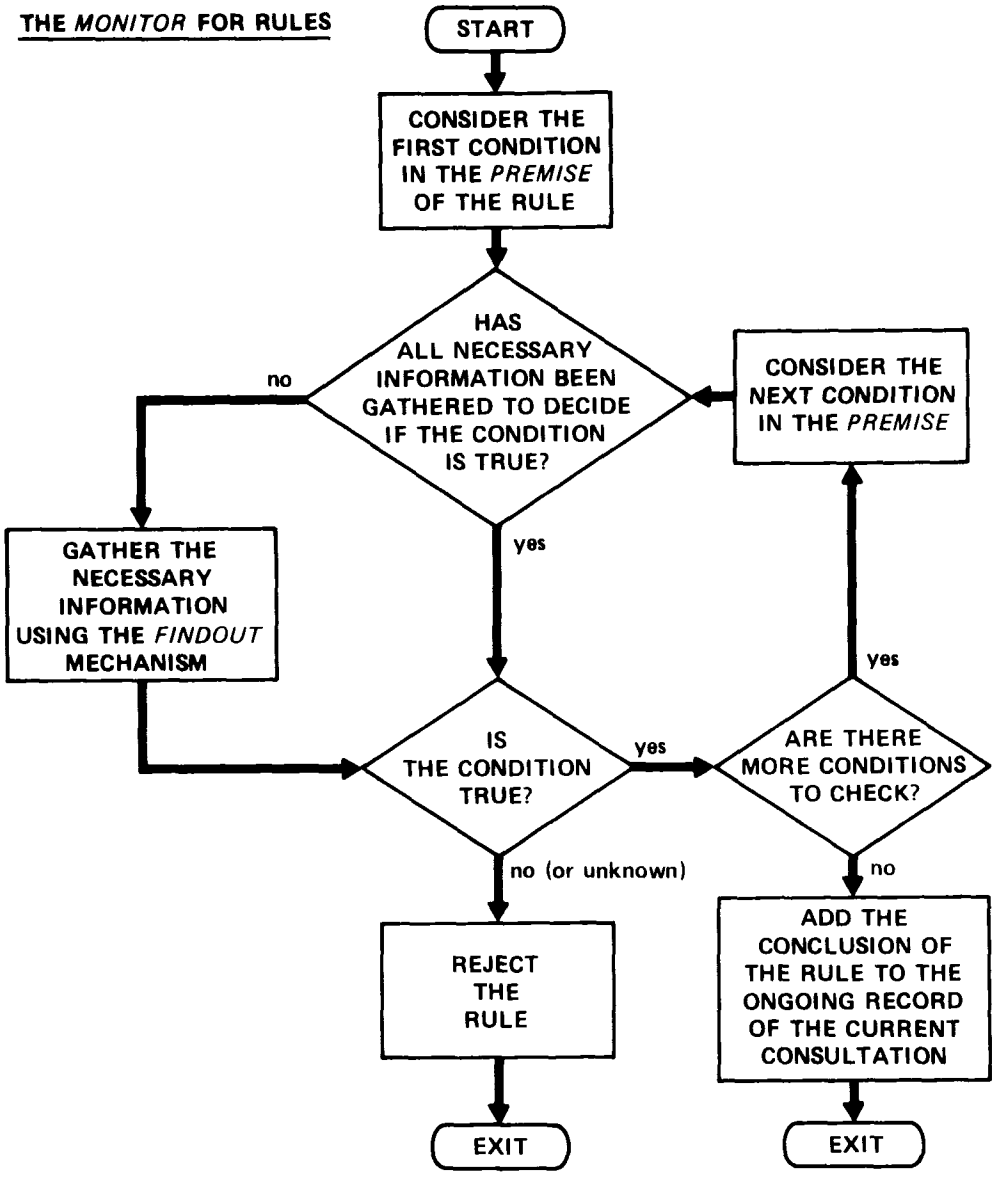

FIGURE 5-6 Flow chart describing how the MONITOR analyzes a rule and decides whether or not it applies in the clinical situation under consideration. Each condition in the premise of the rule references some clinical parameter, and all such conditions must be true for the rule to be accepted (Shortliffe et al., 1975).

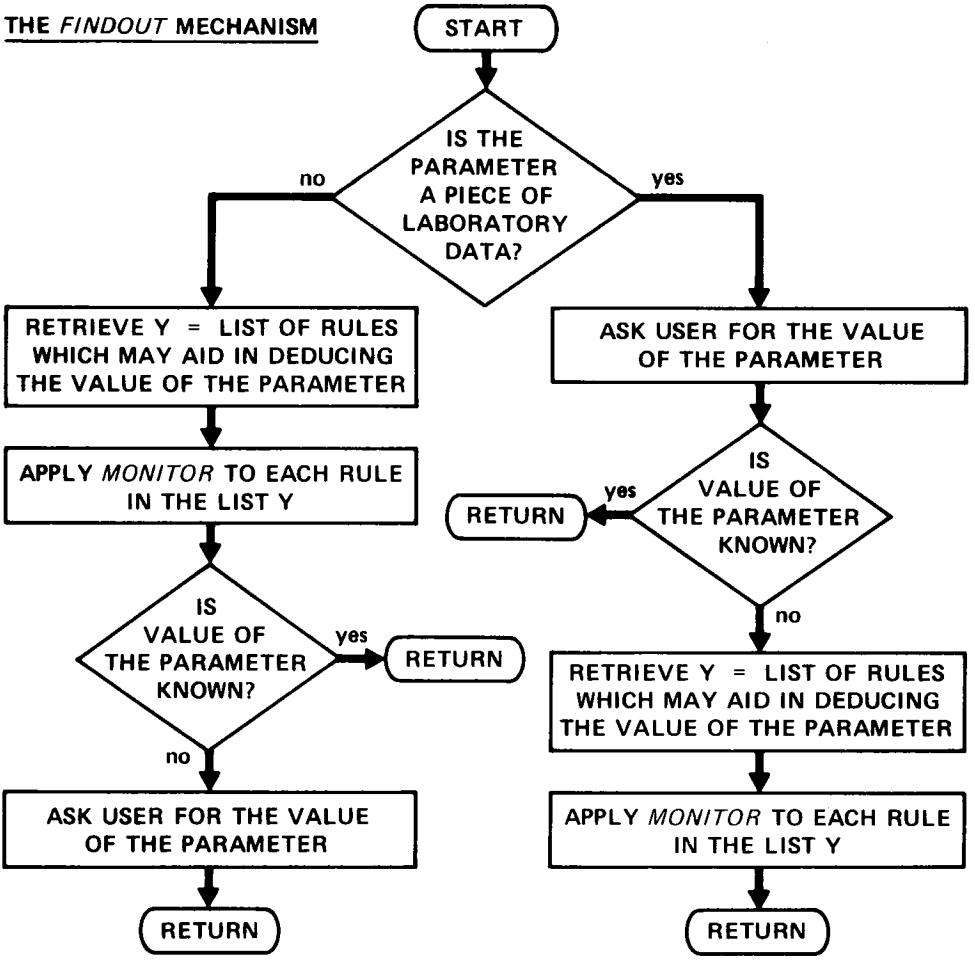

FIGURE 5-7 Flow chart describing the strategy for determining which questions to ask the physician. The derivation of values of parameters may require recursive calls to the MONITOR, thus dynamically creating a reasoning chain specific to the patient under consideration (Shortliffe et al., 1975).

its decision process (Figure 5-7). Thus IDENT is marked as being LABDATA in Figure 5-2.

Recall that the UPDATED-BY property is a list of all rules in the system that permit an inference to be made regarding the value of the indicated parameter. Thus UPDATED-BY is precisely the list called Y in Figure 5-7. Every time a new rule is added to MYCIN's knowledge base, the name of the rule is added to the UPDATED-BY property of the clinical param-

eter referenced in its action or else clause. Thus the new rule immediately becomes available to FINDOUT at times when it may be useful. It is not necessary to specify explicitly its interrelationships with other rules in the system.

Note that FINDOUT is accessed from the MONITOR, but the MONITOR may also be accessed from FINDOUT. This recursion allows self-propagation of a reasoning network appropriate for the patient under consideration and selects only the necessary questions and rules. The first rule passed to the MONITOR is always the goal rule. Since the first condition in the premise of this rule references a clinical parameter named TREATFOR, and since the value of TREATFOR is of course unknown before any data have been gathered, the MONITOR asks FINDOUT to trace the value of TREATFOR. This clinical parameter is not LABDATA, so FINDOUT takes the left-hand pathway in Figure 5-7 and sets Y to the UPDATED-BY property of TREATFOR, the two-element list (RULE090 RULE149). The MONITOR is then called again with RULE090 as the rule for consideration, and FINDOUT is used to trace the values of clinical parameters referenced in the premise of RULE090. Note that this process parallels the informal paraphrase of MYCIN's reasoning given above.

It is important to recognize that FINDOUT does not check to see whether the premise condition is true. Instead, the FINDOUT mechanism traces the clinical parameter *exhaustively* and returns its value to the MONITOR, where the conditional expression may then be evaluated.[4] Hence FINDOUT is called one time at most for a clinical parameter (in a given context—see Section 5.3). When FINDOUT returns a value to the MONITOR, it marks the clinical parameter as having been traced. Thus when the MONITOR reaches the question "HAS ALL NECESSARY INFORMATION BEEN GATHERED TO DECIDE IF THE CONDITION IS TRUE?" (Figure 5-6), the parameter is immediately passed to FINDOUT unless it has been previously marked as traced.

Figure 5-8 is a portion of MYCIN's initial reasoning chain. In Figure 5-8 the clinical parameters being traced are underlined. Thus REGIMEN is the top goal of the system (i.e., it is the clinical parameter in the action clause of the goal rule). Below each parameter are the rules (from the UPDATED-BY property) that may be used for inferring the parameter's value. Clinical parameters referenced in the premise of each of these rules are then listed at the next level in the reasoning network. Rules with multiple premise conditions have their links numbered in accordance with the order in which the parameters are traced (by FINDOUT). ASK1 indicates that a parameter is LABDATA, so its value is automatically asked of the user when it is needed. ASK2 refers to parameters that are not LABDATA but for which no inference rules currently exist, e.g., if the dose of a drug is adequate. One of the goals in the future development of MYCIN's knowl-

[4]The process is slightly different for multi-valued parameters; see Section 5.2.1.

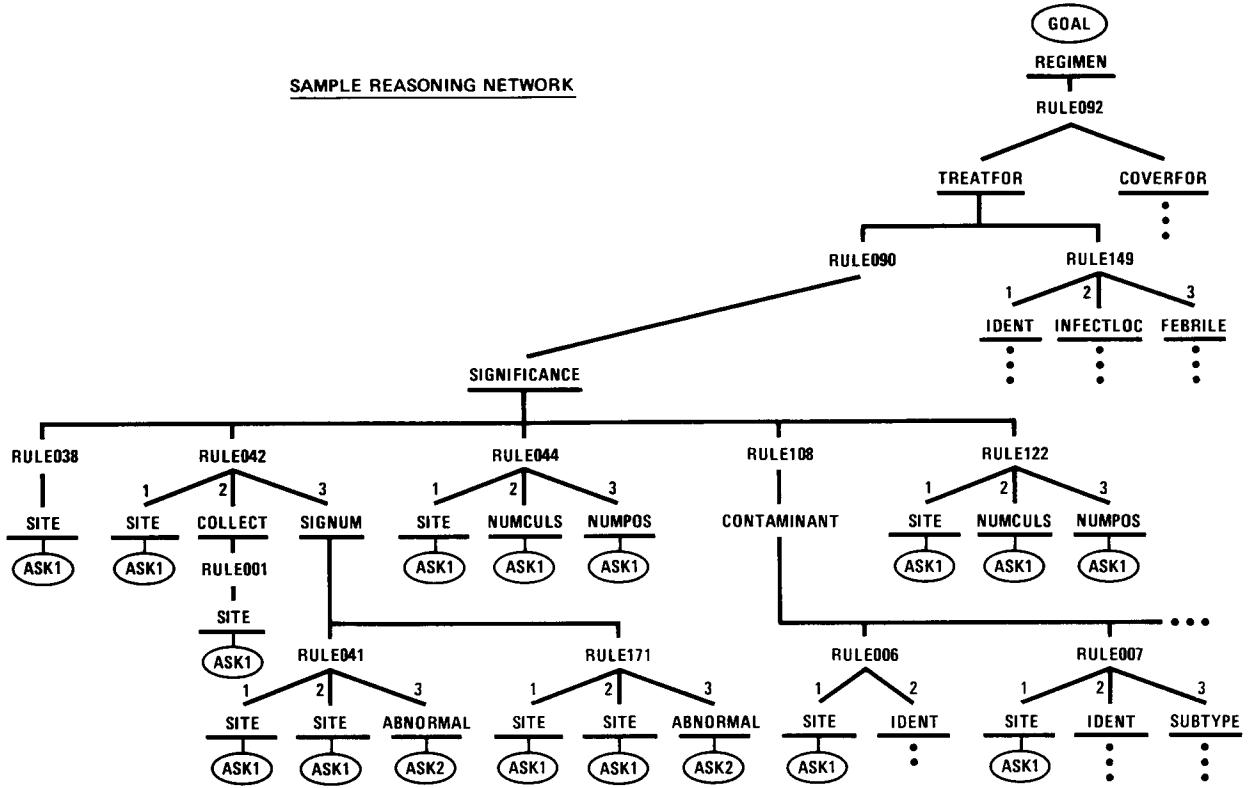

FIGURE 5-8 An example of the kind of reasoning network generated by the MONITOR and FINDOUT mechanisms. Names of clinical parameters are underlined. When a rule has multiple conditions in its premise, numbers have been included to specify the positions of the associated clinical parameters within the premise conditions.

edge base is to acquire enough rules allowing the values of non-LABDATA parameters to be inferred so that ASK2 questions need no longer occur.

Note that the reasoning network in Figure 5-8 is drawn to reflect maximum size. In reality many portions of such a network need not be considered. For example, RULE042 (one of the UPDATED-BY rules under SIGNIFICANCE) is rejected if the SITE condition is found to be false by the MONITOR. When that happens, neither COLLECT nor SIGNUM needs to be traced by FINDOUT, and those portions of the reasoning network are not created. Thus the order of conditions within a premise is highly important. In general, conditions referencing the most common parameters (i.e., those that appear in the premises of the most rules) are put first in the premises of new rules to act as an effective screening mechanism.

A final comment is necessary regarding the box labeled "REJECT THE RULE" in Figure 5-6. This step in the MONITOR actually must check to see if the rule has an else clause. If so, and if the premise is known to be false, the conclusion indicated by the else clause is drawn. If there is no else clause, or if the truth status of the premise is uncertain (e.g., the user has entered UNKNOWN when asked the value of one of the relevant parameters), the rule is simply ignored without any conclusion having been reached.

Asking Questions of the User

The conventions for communication between a program and a physician are a primary factor determining the system's acceptability. We have therefore designed a number of features intended to simplify the interactive process that occurs when FINDOUT reaches one of the boxes entitled "ASK USER FOR THE VALUE OF THE PARAMETER" (Figure 5-7).

When MYCIN requests the value of a single-valued or yes-no parameter, it uses the PROMPT property of the parameter. The user's response is then compared with the EXPECT property of the parameter. If the answer is one of the expected responses, the program simply continues through the reasoning network. Otherwise, MYCIN checks the system dictionary to see if the user's response is a synonym for one of the recognized answers. If this attempt also fails, MYCIN uses Interlisp spelling-correction routines (Teitelman, 1974) to see if a simple spelling or typographical error will account for the unrecognized response. If so, the program makes the correction, prints its assumption, and proceeds as though the user had made no error. If none of these mechanisms succeeds, MYCIN tells the user that the response is not recognized, displays a list of sample responses, and asks the question again.

Multi-valued parameters are handled somewhat differently. FINDOUT recursively traces such parameters in the normal fashion, but when forced to ask a question of the user, it customizes its question to the con-

dition being evaluated in the MONITOR. Suppose, for example, the MONITOR were evaluating the condition (SAME CNTXT INFECT MENINGITIS), i.e., "Meningitis is an infectious disease diagnosis for the patient." If FINDOUT were to ask the question using the regular PROMPT strategy, it would request:

What is the infectious disease diagnosis for PATIENT-1?

The problem is that the patient may have several diagnoses, each of which can be expressed in a variety of ways. If the physician were to respond:

A meningeal inflammation that is probably of infectious origin

MYCIN would be forced to try to recognize that this answer implies meningitis. Our solution has been to customize questions for multi-valued parameters to reflect the value being checked in the current premise condition. The PROMPT1 property is used, and questions always expect a yes or no response:

Is there evidence that the patient has a meningitis?

The advantages of this approach are the resulting ability to avoid natural language processing during the consultation itself and the posing of questions that are specific to the patient under consideration.

In addition to the automatic spelling-correction capability described above, there are a number of options that may be utilized whenever MYCIN asks the user a question:

UNKNOWN	Used to indicate that the physician does not know the answer to the question, usually because the data are unavailable (may be abbreviated U or UNK)
?	Used to request a list of sample recognized responses
??	Used to request a list of *all* recognized responses
RULE	Used to request that MYCIN display the translation of the current decision rule. FINDOUT simply translates the rule being considered by the MONITOR. This feature provides a simple capability for explaining why the program is asking the question. However, it cannot explain motivation beyond the current decision rule.
QA	Used to digress temporarily in order to use the Explanation System. The features of this system are explained in Chapter 18.
WHY	Used to request a detailed explanation of the question being asked. This feature is much more conversational than the RULE option above and permits investigation of the current state of the entire reasoning chain.

CHANGE ###	Used to change the answer to a previous question. Whenever MYCIN asks a question, it prints a number in front of the prompt. Thus CHANGE 4 means "Go back and let me reanswer question 4." The complexities involved in this process are discussed below.
STOP	Halts the program without completing the consultation
HELP	Prints this list

5.2.2 Creation of the Dynamic Data Base

The Consultation System maintains an ongoing record of the consultation. These dynamic data include information entered by the user, inferences drawn using decision rules, and record-keeping data structures that facilitate question answering by the Explanation System (Chapter 18).

Data Acquired from the User

Except for questions related to propagation of the context tree, all queries from MYCIN to the physician request the value of a specific clinical parameter for a specific node in the context tree. The FINDOUT mechanism screens the user's response, stores it in MYCIN's dynamic data base, and returns the value to the MONITOR for evaluation of the conditional statement that generated the question in the first place. The physician's response is stored, of course, so that future rules containing conditions referencing the same clinical parameter will not cause the question to be asked a second time.

As has been noted, however, the values of clinical parameters are always stored along with their associated certainty factors. A physician's response must therefore have a CF associated with it. MYCIN's convention is to assume CF = 1 for the response unless the physician explicitly states otherwise. Thus the following exchange:

```
7) Staining characteristics of ORGANISM-1 (gram):
**GRAMNEG
```

results in:

```
Val[ORGANISM-1,GRAM] = ((GRAMNEG 1.0))
```

If, on the other hand, the user is fairly sure of the answer to a question but wants to indicate uncertainty, he or she may enter a certainty factor in parentheses after the response. MYCIN expects the number to be an integer between -10 and $+10$; the program divides the number by 10 to obtain a CF. Using integers simplifies the user's response and also discour-

ages comparisons between the number and a probability measure. Thus the following exchange:

```
8) Enter the identity (genus) of ORGANISM-1:
** ENTEROCOCCUS (8)
```

results in: Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS-GROUP-D .8))

This example also shows how the dictionary is used to put synonyms into standardized form for the patient's data base (i.e., *Enterococcus* is another name for a group-D *Streptococcus*).

A variant of this last example is the user's option to enter multiple responses to a question, as long as each is modified by a CF. For example:

```
13) Did ORGANISM-2 grow in clumps, chains, or pairs?
** CLUMPS (6) CHAINS (3) PAIRS (-8)
```

results in: Val[ORGANISM-2,CONFORM] = ((CLUMPS .6)(CHAINS .3)(PAIRS -.8))

The CF's associated with the parameter values are then used for evaluation of premise conditions as described earlier. Note that the user's freedom to modify answers increases the flexibility of MYCIN's reasoning. Without the CF option, the user might well have responded UNKNOWN to question 13 above. The demonstrated answer, although uncertain, gives MYCIN much more information than would have been provided by a response of UNKNOWN.

Data Inferred by the System

This subsection explains the <conclusion> item from the BNF rule description, i.e., the functions that are used in action or else clauses when a premise has shown that an indicated conclusion may be drawn. There are only three such functions, two of which (CONCLIST and TRANSLIST) reference knowledge tables (Section 5.1.6) but are otherwise dependent on the third, a function called CONCLUDE. CONCLUDE takes five arguments:

CNTXT	The node in the context tree about which the conclusion is being made
PARAM	The clinical parameter whose value is being added to the dynamic data base
VALUE	The inferred value of the clinical parameter
TALLY	The certainty tally for the premise of the rule (see Section 5.1.5)

CF — The certainty factor for the rule as judged by the expert from whom the rule was obtained

The translation of CONCLUDE depends on the size of CF:

$\|CF\| \geq .8$	"There is strongly suggestive evidence that . . ."
$.4 \leq \|CF\| < .8$	"There is suggestive evidence that . . ."
$\|CF\| < .4$	"There is weakly suggestive evidence that . . ."
Computed CF	"There is evidence that . . ."

Thus the following conclusion:

```
(CONCLUDE CNTXT IDENT STREPTOCOCCUS TALLY .7)
```

translates as:

```
There is suggestive evidence (.7) that the identity of the organism is streptococcus
```

If, for example, the rule with this action clause were successfully applied to ORGANISM-1, an organism for which no previous inferences had been made regarding identity, the result would be:

```
Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS X))
```

where X is the product of .7 and TALLY (see Combining Function 4, Chapter 11). Thus the strength of the conclusion reflects both the CF for the rule *and* the extent to which the premise of the rule is believed to be true for ORGANISM-1.

Suppose a second rule were now found that contains a premise true for ORGANISM-1 and that adds additional evidence to the assertion that the organism is a *Streptococcus*. This new evidence somehow has to be combined with the CF (=X) that is already stored for the hypothesis that ORGANISM-1 is a *Streptococcus*. If Y is the CF calculated for the second rule (i.e., the product of the TALLY for that rule and the CF assigned to the rule by the expert), the CF for the hypothesis is updated to Z so that:

```
Val[ORGANISM-1,IDENT] = ((STREPTOCOCCUS Z))
```

where Combining Function 1 gives $Z = X + Y(1 - X)$. This function is justified and discussed in detail in Chapter 11.

Similarly, additional rules leading to alternate hypotheses regarding the identity of ORGANISM-1 may be successfully invoked. The new hypotheses, along with their associated CF's, are simply appended to the list of hypotheses in Val[ORGANISM-1,IDENT]. Note, of course, that the CF's of some hypotheses may be negative, indicating that there is evidence suggesting that the hypothesis is not true. When there is both positive and negative evidence for a hypothesis, Combining Function 1 must be used in a modified form.

A final point to note is that values of parameters are stored identically regardless of whether the information has been inferred or acquired from

the user. The source of a piece of information is maintained in a separate record. It is therefore easy to incorporate new rules that infer values of parameters for which ASK2 questions to the user were once necessary.

Creating an Ongoing Consultation Record

In addition to information provided or inferred regarding nodes in the context tree, MYCIN's dynamic data base contains a record of the consultation session. This record provides the basis for answering questions about the consultation (Chapter 18).

Two general types of records are kept. One type is information about how values of clinical parameters were obtained. If the value was inferred using rules, a record of those inferences is stored with the rules themselves. Thus whenever an action or else clause is executed, MYCIN keeps a record of the details. The second type of record provides a mechanism for explaining why questions were asked. MYCIN maintains a list of questions, their identifying numbers, the clinical parameter and context involved, plus the rule that led to generation of the question. This information is useful when the user retrospectively requests an explanation for a previous question (Chapter 18).

5.2.3 Self-Referencing Rules

As new rules were acquired from the collaborating experts, it became apparent that MYCIN would need a small number of rules that departed from the strict modularity to which we had otherwise been able to adhere. For example, one expert indicated that he would tend to ask about the typical *Pseudomonas*-type skin lesions only if he already had reason to believe that the organism was a *Pseudomonas*. If the lesions were then said to be evident, however, his belief that the organism was a *Pseudomonas* would be increased even more. A rule reflecting this fact must somehow imply an orderedness of rule invocation; i.e., "Don't try this rule until you have already traced the identity of the organism by using other rules in the system." Our solution has been to reference the clinical parameter early in the premise of the rule as well as in the action, for example:

RULE040

IF: 1) The site of the culture is blood, and
2) The identity of the organism may be pseudomonas, and
3) The patient has ecthyma gangrenosum skin lesions
THEN: There is strongly suggestive evidence (.8) that the identity of the organism is pseudomonas

Note that RULE040 is thus a member of both the LOOKAHEAD property and the UPDATED-BY property for the clinical parameter IDENT. Rules

having the same parameter in both premise and action are termed *self-referencing rules*. The ordered invocation of such rules is accomplished by a generalized procedure described below.

As discussed in Section 5.2.1, a rule such as RULE040 is originally invoked because MYCIN is trying to infer the identity of an organism; i.e., FINDOUT is asked to trace the parameter IDENT and recursively sends the UPDATED-BY list for that parameter to the MONITOR. When the MONITOR reaches RULE040, however, the second premise condition references the same clinical parameter currently being traced by FINDOUT. If the MONITOR merely passed IDENT to FINDOUT again (as called for by the simplified flow chart in Figure 5-6), FINDOUT would begin tracing IDENT for a second time, RULE040 would be passed to the MONITOR yet again, and an infinite loop would occur.

The solution to this problem is to let FINDOUT screen the list called Y in Figure 5-7, i.e., the UPDATED-BY property for the parameter it is about to trace. Y is partitioned by FINDOUT into regular rules and self-referencing rules (where the latter category is defined as those rules that also occur on the LOOKAHEAD list for the clinical parameter). FINDOUT passes the first group of rules to the MONITOR in the normal fashion. After all these rules have been tried, FINDOUT marks the parameter as having been traced and then passes the self-referencing rules to the MONITOR. In this way, when the MONITOR considers the second condition in the premise of RULE040, the condition is evaluated without a call to FINDOUT because the parameter has already been marked as traced. Thus the truth of the premise of a self-referencing rule is determined on the basis of the set of non–self-referencing rules, which were evaluated first. If one of the regular rules permitted MYCIN to conclude that an organism might be a *Pseudomonas*, RULE040 might well succeed when passed to the MONITOR. This mechanism for handling self-referencing rules satisfies the intention of an expert when he or she gives us decision criteria in self-referencing form.

It should be noted that this approach minimizes the potential for self-referencing rules to destroy certainty factor commutativity. By holding these rules until last, we insure that the certainty tally for any of their premises (see Section 5.1.5) is the same regardless of the order in which the non–self-referencing rules were executed. If there is more than one self-referencing rule successfully executed for a given context and parameter, however, the order of their invocation may affect the final CF. The approach we have implemented thus seeks merely to minimize the potential undesirable effects of self-referencing rules.

5.2.4 Preventing Reasoning Loops

Self-referencing rules are actually a special case of a more general problem. Reasoning loops involving multiple rules cannot be handled by the mechanism described above. The difference is that self-referencing rules are

intentional parts of MYCIN's knowledge base whereas reasoning loops are artifacts that must somehow be avoided.

For the following discussion we introduce the following notation:

$$[q] \quad X ::> Y$$

means that decision rule [q] uses clinical parameter X to reach a conclusion regarding the value of clinical parameter Y. Thus a self-referencing rule may be represented by:

$$[a] \quad E ::> E$$

where E is the clinical parameter that is referenced in both the premise and the action of the rule. Consider now the following set of rules:

$$[1] \quad A ::> B$$

$$[2] \quad B ::> C$$

$$[3] \quad C ::> D$$

$$[4] \quad D ::> A$$

Rule [1], for example, says that under certain unspecified conditions, the value of A can be used to infer the value of B. Now suppose that the MONITOR asks FINDOUT to trace the clinical parameter D. Then MYCIN's recursive mechanism would create the following reasoning chain:

$$\ldots D \overset{[4]}{::>} A \overset{[1]}{::>} B \overset{[2]}{::>} C \overset{[3]}{::>} D$$

The difference between this looped reasoning chain and a self-referencing rule is that Rule [4] was provided as a mechanism for deducing the value of A, not for reinforcing the system's belief in the value of D. In cases where the value of A is of primary interest, the use of Rule [4] would be appropriate.

MYCIN solves this problem by keeping track of all parameters currently being traced by the FINDOUT mechanism. The MONITOR then simply ignores a rule if one of the parameters checked in its premise is already being traced. The result, with the value of D as the goal, is a three-membered reasoning chain in the case above:

$$A \overset{[1]}{::>} B \overset{[2]}{::>} C \overset{[3]}{::>} D$$

Rule [4] is rejected because parameter D is already being traced elsewhere in the current reasoning chain. If the value of A were the main goal, however, the chain would be

```
      [2]     [3]     [4]
B ::> C ::> D ::> A
```

Note that this simple mechanism allows us to have potential reasoning loops in the knowledge base but to select only the relevant nonlooping portions for consideration of a given patient.

A similar problem can occur when a rule permits two conclusions to be made, each about a different clinical parameter. MYCIN prevents loops in such circumstances by refusing to permit the same rule to occur twice in the current reasoning chain.

5.3 Propagation of the Context Tree

The mechanism by which the context tree is customized for a given patient has not yet been discussed. As described in Section 5.2.2, the consultation system begins simply by creating the patient context and then attempting to execute the goal rule. All additional nodes in the context tree are thus added automatically during the unwinding of MYCIN's reasoning regarding the premise of the goal rule. This section first explains the data structures used for creating new nodes. Mechanisms for deciding when new nodes should be added are then discussed.

5.3.1 Data Structures Used for Sprouting Branches

Section 5.1.2 was devoted to an explanation of the context tree. At that time we described the different kinds of contexts and explained that each node in the tree is an instantiation of the appropriate context-type. Each context-type is characterized by the following properties:

PROMPT1	A sentence used to ask the user whether the first node of this type should be added to the context tree; expects a yes-no answer
PROMPT2	A sentence used to ask the user whether subsequent nodes of this type should be added to the context tree
PROMPT3	Replaces PROMPT1 when it is used. This is a message to be printed out if MYCIN assumes that there is at least one node of this type in the tree.
PROPTYPE	Indicates the category of clinical parameters (see Section 5.1.3) that may be used to characterize a context of this type

SUBJECT	Indicates the categories of rules that may be applied to a context of this type
SYN	Indicates a conversational synonym for referring to a context of this type. MYCIN uses SYN when filling in the asterisk of PROMPT properties for clinical parameters.
TRANS	Used for English translations of rules referencing this type of context
TYPE	Indicates what kind of internal name to give a context of this type
MAINPROPS	Lists the clinical parameters, if any, that are to be automatically traced (by FINDOUT) whenever a context of this type is created
ASSOCWITH	Gives the context-type of nodes in the tree immediately above contexts of this type

Two sample context-types are shown in Figure 5-9. The following observations may help clarify the information given in that figure:

1. PRIORCULS: Whenever a prior culture is created, it is given the name CULTURE-# (see TYPE), where # is the next unassigned culture number. The values of SITE and WHENCUL are immediately traced using the FINDOUT mechanism (see MAINPROPS). The culture node is put in the context tree below a node of type PERSON (see ASSOCWITH), and the new context may be characterized by clinical parameters of the type PROP-CUL (see PROPTYPE). The prior culture may be the context for either PRCULRULES or CULRULES (see SUBJECT) and is translated, in questions to the user, as "this (site) culture" (see SYN) where (site) is replaced by the site of the culture if it is known.
2. CURORG: Since there is a PROMPT3 rather than a PROMPT1, MYCIN prints out the PROMPT3 message and assumes (without asking) that there is at least one CURORG for each CURCUL (see ASSOCWITH); the other CURORG properties correspond to those described above for PRIORCULS.

Whenever MYCIN creates a new context using these models, it prints out the name of the new node in the tree, e.g.:

------ORGANISM-1------

Thus the user is familiar with MYCIN's internal names for the cultures, organisms, and drugs under discussion. The node names may then be used in MYCIN's questions at times when there may be ambiguity regarding which node is the current context, e.g.:

Is the patient's illness with the staphylococcus (ORGANISM-2) a hospital-acquired infection?

PRIORCULS
ASSOCWITH: PERSON
MAINPROPS: (SITE WHENCUL)
PROMPT1: (Were any organisms that were significant (but no longer require therapeutic attention) isolated within the last approximately 30 days?)
PROMPT2: (Any other significant earlier cultures from which pathogens were isolated?)
PROPTYPE: PROP-CUL
SUBJECT: (PRCULRULES CULRULES)
SYN: (SITE (this * culture))
TRANS: (PRIOR CULTURES OF *)
TYPE: CULTURE-

CURORG
ASSOCWITH: CURCUL
MAINPROPS: (IDENT GRAM MORPH SENSITIVS)
PROMPT2: (Any other organisms isolated from * for which you would like a therapeutic recommendation?)
PROMPT3: (I will refer to the first offending organism from * as:)
PROPTYPE: PROP-ORG
SUBJECT: (ORGRULES CURORGRULES)
SYN: (IDENT (the *))
TRANS: (CURRENT ORGANISMS OF *)
TYPE: ORGANISM-

FIGURE 5-9 *Context trees such as that shown in Figure 5-1 are generated from prototype context-types such as those shown here. The defining properties are described in the text.*

It should also be noted that when PROMPT1 or PROMPT2 is used to ask a question, the physician need not be aware that the situation is different from that occurring when FINDOUT asks questions. All the user options described in Section 5.2.1 operate in the normal fashion.

Finally, the MAINPROPS property (later called INITIALDATA) requires brief explanation. The claim was previously made that clinical parameters are traced and their values requested by FINDOUT only when they are needed for evaluation of a rule that has been invoked. Yet we must now acknowledge that certain LABDATA parameters are automatically traced whenever a node for the context tree is created. The reason for this departure is an attempt to keep the program acceptable to physicians. Since the order of rules on UPDATED-BY lists is arbitrary, the order in which questions are asked is somewhat arbitrary as well. We have found that physicians are annoyed if the "basic" questions are not asked first, as soon as the context is created. The MAINPROPS convention forces certain

standard questions early in the characterization of a node in the context tree. Parameters not on the MAINPROPS list are then traced in an arbitrary order that depends on the order in which rules are invoked. Since the parameters on MAINPROPS lists are important pieces of information that would uniformly be traced by FINDOUT anyway, the convention we have implemented forces a standardized ordering of the "basic" questions without generating useless information.

5.3.2 Explicit Mechanisms for Branching

There are two situations under which MYCIN attempts to add new nodes to the context tree. The simpler case occurs when rules explicitly reference contexts that have not yet been created. Suppose, for example, MYCIN is trying to determine the identity of a current organism and therefore invokes the following CURORGRULE:

IF: 1) The identity of the organism is not known with certainty, and
2) This current organism and prior organisms of the patient agree with respect to the following properties: GRAM MORPH
THEN: There is weakly suggestive evidence that each of them is a prior organism with the same identity as this current organism

The second condition in the premise of this rule references other nodes in the tree, namely nodes of the type PRIORORGS. If no such nodes exist, the MONITOR asks FINDOUT to trace PRIORORGS in the normal fashion. The difference is that PRIORORGS is not a clinical parameter but a context-type. FINDOUT therefore uses PROMPT1 of PRIORORGS to ask the user if there is at least one organism. If so, an instantiation of PRIORORGS is added to the context tree, and its MAINPROPS are traced. PROMPT2 is then used to see if there are any additional prior organisms, and the procedure continues until the user indicates there are no more PRIORORGS that merit discussion. Finally, FINDOUT returns the list of prior organisms to the MONITOR so that the second condition in the rule above can be evaluated.

5.3.3 Implicit Mechanisms for Branching

There are two kinds of implicit branching mechanisms. One of these is closely associated with the example of the preceding section. As shown in Figure 5-1, a prior organism is associated with a prior culture. But the explicit reference to prior organisms in the rule above made no mention of prior cultures. Thus if FINDOUT tries to create a PRIORORGS in

response to an explicit reference but finds there are no PRIORCULS, the program knows there is an implied need to ask the user about prior cultures before asking about prior organisms. Since PRIORCULS are associated with the patient, and since the patient node already exists in the context tree, only one level of implicit branching is required in the evaluation of the rule.

The other kind of implicit branching occurs when the MONITOR attempts to evaluate a rule for which no appropriate context exists. For example, the first rule invoked in an effort to execute the goal rule is a CURORGRULE (see RULE090, Figure 5-8). Since no current organism has been created at the time the MONITOR is passed this CURORGRULE, MYCIN automatically attempts to create the appropriate nodes and then to apply the invoked rule to each.

5.4 Selection of Therapy

The preceding discussion concentrated on the premise of MYCIN's principal goal rule (RULE092). This section explains what happens when the premise is found to be true and the two-step action clause is executed. Unlike other rules in the system, the goal rule does not lead to a conclusion (Section 5.2.2) but instead instigates actions. The functions in the action of the goal rule thus correspond to the <actfunc> class that was introduced in the BNF description. The first of these functions causes a list of potential therapies to be created. The second allows the best drug or drugs to be selected from the list of possibilities.

5.4.1 Creation of the Potential Therapy List

There is a class of decision rules, the THERULES, that are never invoked by MYCIN's regular control structure because they do not occur on the UPDATED-BY list of any clinical parameter. These rules contain sensitivity information for the various organisms known to the system, for example:

IF: *The identity of the organism is pseudomonas*
THEN: I recommend therapy chosen from among the following drugs:
1 - colistin (.98)
2 - polymyxin (.96)
3 - gentamicin (.96)
4 - carbenicillin (.65)
5 - sulfisoxazole (.64)

The numbers associated with each drug are the probabilities that a *Pseudomonas* isolated at Stanford Hospital will be sensitive (*in vitro*) to the in-

dicated drug. The sensitivity data were acquired from Stanford's microbiology laboratory (and could easily be adjusted to reflect changing resistance patterns at Stanford or the data for some other hospital desiring a version of MYCIN with local sensitivity information). Rules such as the one shown here provide the basis for creating a list of potential therapies. There is one such rule for every kind of organism known to the system.

MYCIN selects drugs only on the basis of the identity of offending organisms. Thus the program's first task is to decide, for each current organism deemed to be significant, which hypotheses regarding the organism's identity (IDENT) are sufficiently likely that they must be considered in choosing therapy. MYCIN uses the CF's of the various hypotheses in order to select the most likely identities. Each identity is then given an *item number* (see below) and the process is repeated for each significant current organism. The *Set of Indications* for therapy is then printed out, e.g.:

My therapy recommendation will be based on the following possible identities of the organism(s) that seem to be significant:

<Item 1> The identity of ORGANISM-1 may be STREPTOCOCCUS-GROUP-D
<Item 2> The identity of ORGANISM-1 may be STREPTOCOCCUS-ALPHA
<Item 3> The identity of ORGANISM-2 is PSEUDOMONAS

Each item in this list of therapy indications corresponds to one of the THERULES. Thus MYCIN retrieves the list of potential therapies for each indication from the associated THERULE. The default (*in vitro*) statistical data are also retrieved. MYCIN then replaces the default sensitivity data with real data about those of the patient's organisms, if any, for which actual sensitivity information is available from the laboratory. Furthermore, if MYCIN has inferred sensitivity information from the *in vivo* performance of a drug that has already been administered to the patient, this information also replaces the default sensitivity data. Thus the compiled list of potential therapies is actually several lists, one for each item in the Set of Indications. Each list contains the names of drugs and, in addition, the associated numbers representing MYCIN's judgment regarding the organism's sensitivity to each of the drugs.

5.4.2 Selecting the Preferred Drug from the List

When MYCIN recommends therapy, it tries to suggest a drug for each of the items in the Set of Indications. Thus the problem reduces to selecting the best drug from the therapy list associated with each item. Clearly, the probability that an organism will be sensitive to a drug is an important factor in this selection process. However, there are several other consid-

erations. MYCIN's strategy is to select the best drug on the basis of sensitivity information but then to consider contraindications for that drug. Only if a drug survives this second screening step is it actually recommended. Furthermore, MYCIN also looks for ways to minimize the number of drugs recommended and thus seeks therapies that cover for more than one of the items in the Set of Indications. The selection/screening process is described in the following two subsections.

Choosing the Apparent First-Choice Drug

The procedure used for selecting the apparent first-choice drug is a complex algorithm that is somewhat arbitrary and is thus currently (1974) under revision. This section describes the procedure in somewhat general terms since the actual LISP functions and data structures are not particularly enlightening.

There are three initial considerations used in selecting the best therapy for a given item:

1. the probability that the organism is sensitive to the drug;
2. whether the drug is already being administered;
3. the relative efficacy of drugs that are otherwise equally supported by the first two criteria.

As is the case with human consultants, MYCIN does not insist on a change in therapy if the physician has already begun a drug that may work, even if that drug would not otherwise be MYCIN's first choice. Drugs with sensitivity numbers within .05 of one another are considered to be almost identical on the basis of the first criterion. Thus the rule in the previous section, for example, indicates no clear preference among colistin, polymyxin, and gentamicin[5] for *Pseudomonas* infections (if default sensitivity information from the rule is used). However, our collaborating experts have ranked the relative efficacy of antimicrobials on a scale from 1 to 10. The number reflects such factors as whether the drug is bacteriostatic or bacteriocidal or its tendency to cause allergic sensitization. Since gentamicin has a higher relative efficacy than either colistin or polymyxin, it is the first drug considered for *Pseudomonas* infections (unless known sensitivity information or previous drug experience indicates that an alternate choice is preferable).

Once MYCIN has selected the apparent best drug for each item in the Set of Indications, it checks to see if one of the drugs is also useful for one or more of the other indications. For example, if the first-choice drug for

[5]*Ed. note:* Amikacin and tobramycin were not yet available in 1974 when this rule was written. The knowledge base was later updated with the new drug information.

Item 1 is the second-choice drug for Item 2 and if the second-choice drug for Item 2 is almost as strongly supported as the first-choice drug, Item 1's first-choice drug also becomes Item 2's first-choice drug. This strategy permits MYCIN to attempt to minimize the number of drugs to be recommended.

A similar strategy is used to avoid giving two drugs of the same drug class. For example, MYCIN knows that if the first choice for one item is penicillin and the first choice for another is ampicillin, then the ampicillin may be given for both indications (because ampicillin covers essentially all organisms sensitive to penicillin).

In the ideal case MYCIN will find a single drug that effectively covers for all the items in the Set of Indications. But even if each item remains associated with a different drug, a screening stage to look for contraindications is required. This rule-based process is described in the next subsection. It should be stressed, however, that the manipulation of drug lists described above is algorithmic; i.e., it is coded in LISP functions that are called from the action clause of the goal rule. There is considerable "knowledge" in this process. Since rule-based knowledge provides the foundation of MYCIN's ability to explain its decisions, it would be desirable eventually to remove this therapy selection method from functions and place it in decision rules.[6]

Rule-Based Screening for Contraindications

Unlike the complex list manipulations described in the preceding subsection, criteria for ruling out drugs under consideration may be effectively placed in rules. The rules in MYCIN for this purpose are termed ORDERRULES. A sample rule of this type is:

IF: 1) The therapy under consideration is tetracycline, and
2) The age (in years) of the patient is less than 13
THEN: There is strongly suggestive evidence (.8) that tetracycline is not a potential therapy for use against the organism

In order to use MONITOR and FINDOUT with such rules, we must construct appropriate nodes in the context tree and must be able to characterize them with clinical parameters. The context-type used for this purpose is termed POSSTHER and the parameters are classified as PROPTHER. Thus when MYCIN has selected the apparent best drugs for the items in the Set of Indications, it creates a context corresponding to each of these drugs. POSSTHER contexts occur below CURORGS in the context tree. FINDOUT is then called to trace the relevant clinical parameter,

[6]*Ed. note:* See the next chapter for a discussion of how this was later accomplished.

which collects contraindication information (i.e., this becomes a new goal statement), and the normal recursive mechanism through the MONITOR insures that the proper ORDERRULES are invoked.

ORDERRULES allow a great deal of drug-specific knowledge to be stored. For example, the rule above insures that tetracycline is ruled out in youngsters who still have developing bone and teeth.[7] Similar rules tell MYCIN never to give streptomycin or carbenicillin alone, not to give sulfonamides except in urinary tract infections, and not to give cephalothin, clindamycin, lincomycin, vancomycin, cefazolin, or erythromycin if the patient has meningitis. Other ORDERRULES allow MYCIN to consider the patient's drug allergies, dosage modifications, or ecological considerations (e.g., save gentamicin for *Pseudomonas, Serratia,* and *Hafnia* unless the patient is so sick that you cannot risk using a different aminoglycoside while awaiting lab sensitivity data). Finally, there are rules that suggest appropriate combination therapies (e.g., add carbenicillin to gentamicin for known *Pseudomonas* infections). In considering such rules MYCIN often is forced to ask questions that never arose during the initial portion of the consultation. Thus the physician is asked additional questions during the period after MYCIN has displayed the items in the Set of Indications but before any therapy is actually recommended.

After the presumed first-choice drugs have been exposed to the ORDERRULE screening process, MYCIN checks to see whether any of the drugs is now contraindicated. If so, the drug-ranking process is repeated. New first-choice drugs are then subjected to the ORDERRULES. The process continues until all the first-choice drugs have been instantiated as POSSTHERS. These then become the system's recommendations. Note that this strategy may result in the recommendation of drugs that are only mildly contraindicated so long as they are otherwise strongly favored. The therapy recommendation itself takes the following form:

My preferred therapy recommendation is as follows:
In order to cover for Items <1> <2> <3>:
Give the following in combination:
1. PENICILLIN
   Dose: 285,000 UNITS/KG/DAY - IV
2. GENTAMICIN
   Dose: 1.7 MG/KG Q8H - IV OR IM
   Comments: MODIFY DOSE IN RENAL FAILURE

The user may also ask for second, third, and subsequent therapy recommendations until MYCIN is able to suggest no reasonable alternatives. The mechanism for these iterations is merely a repeat of the processes described above but with recommended drugs removed from consideration.

---

[7] *Ed. note:* This rule ignores any statement of the mechanism whereby its conclusion follows from its premise. The lack of underlying "support" knowledge accounts for changes introduced in GUIDON when MYCIN's rules were used for education. See Part Eight for further discussion of this point.

5.5 Mechanisms for Storage of Patient Data

5.5.1 Changing Answers to Questions

If a physician decides he or she wants to change a response to a question that has already been answered, MYCIN must do more than merely redisplay the prompt, accept the user's new answer, and make the appropriate change to the value of the clinical parameter in question. In general, the question was originally asked because the premise of a decision rule referenced the clinical parameter. Thus the original response affected the evaluation of at least one rule, and subsequent pathways in the reasoning network may have been affected as well. It is therefore necessary for MYCIN somehow to return to the state it was in at the time the question was originally asked. Its subsequent actions can then be determined by the corrected user response.

Reversing all decisions made since a question was asked is a complex problem, however. The most difficult task is to determine what portions of a parameter's cumulative CF preceded or followed the question requiring alteration. In fact, the extra data structures needed to permit this kind of backing up are so large and complicated, and would be used so seldom, that it seems preferable simply to restart the consultation from the beginning when the user wants to change one of his or her answers.

Restarting is of course also less than optimal, particularly if it requires that the physician reenter the answers to questions that were correct the first time around. Our desire to make the program acceptable to physicians required that we devise some mechanism for changing answers, but restarting from scratch also had obvious drawbacks regarding user acceptance of the system. We therefore needed a mechanism for restarting MYCIN's reasoning process but avoiding questions that had already been answered correctly. When FINDOUT asks questions, it therefore uses the following three-step algorithm:

1. Before asking the question, check to see if the answer is already stored (in the Patient Data Table—see Step 3 below); if the answer is there, use that value rather than asking the user; otherwise go to Step 2.
2. Ask the question using PROMPT or PROMPT1 as usual.
3. Store the user's response in the dynamic record of facts about the patient, called the Patient Data Table, under the appropriate clinical parameter and context.

The Patient Data Table, then, is a growing record of the user's responses to questions from MYCIN. It is entirely separate from the dynamic data record that is explicitly associated with the nodes in the context tree. Note

that the Patient Data Table contains only the text responses of the user—there is no CF information (unless included in the user's response), nor are there data derived from MYCIN's rule-based inferences.

The Patient Data Table and the FINDOUT algorithm make the task of changing answers much simpler. The technique MYCIN uses is the following:

a. Whenever the user wants to change the answer to a previous question, he or she enters CHANGE <numbers>, where <numbers> is a list of the questions whose answers need correction.

b. MYCIN looks up the indicated question numbers in its question record.

c. The user's responses to the indicated questions are removed from the current Patient Data Table.

d. MYCIN reinitializes the system, erasing the entire context tree, including all associated parameters; however it leaves the Patient Data Table intact except for the responses deleted in (c).

e. MYCIN restarts the consultation from the beginning.

This simple mechanism results in a restarting of the Consultation System but does not require that the user enter correct answers a second time. Since the Patient Data Table is saved, Step 1 of the FINDOUT algorithm above will find all the user's responses until the first question requiring alteration is reached. Thus the first question asked the user after he or she gives the CHANGE command is, in fact, the earliest of the questions he or she wants to change. There may be a substantial pause after the CHANGE command while MYCIN reasons through the network to the first question requiring alteration, but a pause is to be preferred over a mechanism requiring reentry of all answers. The implemented technique is entirely general because answers to questions regarding context tree propagation are also stored in the Patient Data Table.

5.5.2 Remembering Patients for Future Reference

When a consultation is complete, the Patient Data Table contains all responses necessary for generating a complete consultation for that patient. It is therefore straightforward to store the Patient Data Table (on disk or tape) so that it may be reloaded in the future. FINDOUT will automatically read responses from the table, rather than ask the user, so a consultation may be run several times on the basis of only a single interactive session.

There are two reasons for storing Patient Data Tables for future reference. One is their usefulness in evaluating changes to MYCIN's knowledge base. The other is the resulting ability to reevaluate patients once new clinical information becomes available.

Evaluating New Rules

New rules may have a large effect on the way a given patient case is handled by MYCIN. For example, a single rule may reference a clinical parameter not previously sought or may lead to an entirely new chain in the reasoning network. It is therefore useful to reload Patient Data Tables and run a new version of MYCIN on old patient cases. A few new questions may be asked (because their responses are not stored in the Patient Data Table). Conclusions regarding organism identities may then be observed, as may the program's therapeutic recommendations. Any changes from the decisions reached during the original run (i.e., when the Patient Data Table was created) must be explained. When a new version of MYCIN evaluates several old Patient Data Tables in this manner, aberrant side effects of new rules may be found. Thus a library of stored patient cases provides a useful mechanism for screening new rules before they become an integral part of MYCIN's knowledge base.

Reevaluating Patient Cases

The second use for stored Patient Data Tables is the reevaluation of patient data once additional laboratory or clinical information becomes available. If a user answers several questions with UNKNOWN during the initial consultation session, MYCIN's advice will of course be based on less than complete information. After storing the Patient Data Table, however, the physician may return for another consultation in a day or so once he or she has more specific information. MYCIN can use the previous Patient Data Table for responses to questions whose answers are still up to date. The user therefore needs to answer only those questions that reference new information. A mechanism for the physician to indicate directly what new data are available has not yet been automated, however.[8]

A related capability to be implemented before MYCIN becomes available in the clinical setting is a SAVE command.[9] If a physician must leave the computer terminal midway through a consultation, this option will save the current Patient Data Table on the disk. When the physician returns to complete the consultation, he or she will reload the patient record and the session will continue from the point at which the SAVE command was entered.

It should be stressed that saving the current Patient Data Table is *not* the same as saving the current state of MYCIN's reasoning. Thus, as we have stated above, changes to MYCIN's rule corpus may result in different advice from the same Patient Data Table.

---

[8]*Ed. note:* A RESTART option was subsequently developed to permit reassessment of cases over time.

[9]*Ed. note:* This option was also subsequently implemented.

5.6 Suggested Improvements to the System

This section summarizes some ideas for improvement of the consultation program described in this chapter. Each of the topics mentioned is the subject of current (1974) efforts by one or more of the researchers associated with the MYCIN project.

5.6.1 Dynamic Ordering of Rules

The order in which rules are invoked by the MONITOR is currently controlled solely by their order on the UPDATED-BY property of the clinical parameter being traced.[10] The order of rules on the UPDATED-BY property is also arbitrary, tending to reflect nothing more than the order in which rules were acquired. Since FINDOUT sends all rules on such lists to the MONITOR and since our certainty factor combining function is commutative, the order of rules is unimportant.

Some rules are much more useful than others in tracing the value of a clinical parameter. For example, a rule with a six-condition premise that infers the value of a parameter with a low CF requires a great deal of work (as many as six calls to FINDOUT) with very little gain. On the other hand, a rule with a large CF and only one or two premise conditions may easily provide strong evidence regarding the value of the parameter in question. It may therefore be wise for FINDOUT to order the rules in the UPDATED-BY list on the basis of both information content (CF) and the work necessary to evaluate the premise. Then if the first few rules are successfully executed by the MONITOR, the CF associated with one of the values of the clinical parameter may be so large that invocation of subsequent rules will require more computational effort than they are worth. If FINDOUT therefore ignores such rules (i.e., does not bother to pass them to the MONITOR), considerable time savings may result. Furthermore, entire reasoning chains will in some cases be avoided, and the number of questions asked the user could accordingly be decreased.[11]

5.6.2 Dynamic Ordering of Conditions Within Rules

The MONITOR diagram in Figure 5-6 reveals that conditions are evaluated strictly in the order in which they occur within the premise of the rule. The order of conditions is therefore important, and the most com-

[10] An exception to this point is the self-referencing rules—see Section 5.2.3.

[11] *Ed. note:* Many of these ideas were later implemented and are briefly mentioned in Chapter 4. For example, *meta-rules* provided a mechanism for encoding strategies to help select the most pertinent rules in a set, and the concept of a *unity path* was implemented to favor chains of rules that reached conclusions with certainty at each step in the chain.

monly referenced clinical parameters should be placed earliest in the premise.

Suppose, however, that in a given consultation the clinical parameter referenced in the fourth condition of a rule has already been traced by FINDOUT because it was referenced in some other rule that the MONITOR has already evaluated. As currently designed, MYCIN checks the first three conditions first, even if the fourth condition is already known to be false. Since the first three conditions may well require calls to FINDOUT, the rule may generate unnecessary questions and expand useless reasoning chains.

The solution to this problem would be to redesign the MONITOR so that it reorders the premise conditions, first evaluating those that reference clinical parameters that have already been traced by FINDOUT. In this way a rule will not cause new questions or additions to the reasoning network if any of its conditions are known to be false at the outset.[12]

5.6.3 Prescreening of Rules

An alternate approach to the problem described in the preceding section would be for FINDOUT to judge the implications of every parameter it traces. Once the value has been determined by the normal mechanism, FINDOUT could use the LOOKAHEAD list for the clinical parameter in order to identify all rules referencing the parameter in their premise conditions. FINDOUT could then evaluate the relevant conditions and mark the rule as failing if the condition turns out to be false. Then, whenever the MONITOR begins to evaluate rules that are invoked by the normal recursive mechanism, it will check to see if the rule has previously been marked as false by FINDOUT. If so, the rule could be quickly ruled out without needing to consider the problem of reordering the premise conditions.

At first glance, the dynamic reordering of premise conditions appears to be a better solution than the one just described. The problem with rule prescreening is that it requires consideration of all rules on the parameter's LOOKAHEAD list, some of which may never actually be invoked during the consultation.[13]

5.6.4 Placing All Knowledge in Rules

Although most of MYCIN's knowledge is placed in decision rules, we have pointed out several examples of knowledge that is not rule-based. The simple lists and knowledge tables may be justified on the basis of efficiency,

[12] *Ed. note:* The *preview mechanism* in MYCIN was eventually implemented to deal with this issue.

[13] *Ed. note:* It was for this reason that the idea outlined here was never implemented.

especially since those knowledge structures may be directly accessed by rules.

However, the algorithmic mechanisms for therapy selection are somewhat more bothersome. Although we have managed to put many drug-related decision criteria in the ORDERRULES, the mechanisms for creating the potential therapy lists and for choosing the apparent first-choice drug are programmed explicitly in a series of relatively complex LISP functions. Since MYCIN's ability to explain itself is based on rule retrieval, the system cannot give good descriptions of these drug selection procedures. It is therefore desirable to place more of the drug selection knowledge in rules.

Such efforts should provide a useful basis for evaluating the power of our rule-based formalism. If the goal-oriented control structure we have developed is truly general, one would hope that algorithmic approaches to the construction and ordering of lists could also be placed in decision rule format. We therefore intend to experiment with ways for incorporating the remainder of MYCIN's knowledge into decision rules that are invoked by the standard MONITOR/FINDOUT process.[14]

5.6.5 The Need for a Context Graph

The context tree used by MYCIN is the source of one of the system's primary problems in attempting to simulate the consultation process. Every node in the context tree leads to the uppermost patient node by a single pathway. In reality, however, drugs, patients, organisms, and cultures are not interrelated in this highly structured fashion. For example, drugs are often given to cover for more than one organism. The context tree does not permit a single CURDRUG or PRIORDRUG to be associated with more than a single organism. What we need, therefore, is a network of contexts in the form of a graph rather than a pure tree. The reasons why MYCIN currently needs a tree-structured context network are explained in Section 5.1.2. We have come to recognize that a context graph capability is an important extension of the current system, however, and this will be the subject of future design modifications.[15] When implemented, for example, it will permit a physician to discuss a prior drug only once, even though it may have been given to cover for several prior organisms.

---

[14] *Ed. note:* Rule-based encoding of the therapy selection algorithm was eventually undertaken and is described in the next chapter.

[15] *Ed. note:* This problem was never adequately solved and remains a limitation of the EMYCIN architecture (Part Five). A partial solution was achieved when predicate functions were developed that allowed a specific rule to be applied to all contexts of a given type and to draw inferences in one part of the context tree based on findings elsewhere in the context tree.