PART TWO

Using Rules

3

The Evolution of MYCIN's Rule Form

There is little doubt that the decision to use rules to encode infectious disease knowledge in the nascent MYCIN system was largely influenced by our experience using similar techniques in DENDRAL. However, as mentioned in Chapter 1, we did experiment with a semantic network representation before turning to the production rule model. The impressive published examples of Carbonell's SCHOLAR system (Carbonell, 1970a; 1970b), with its ability to carry on a mixed-initiative dialogue regarding the geography of South America, seemed to us a useful model of the kind of rich interactive environment that would be needed for a system to advise physicians.

Our disenchantment with a pure semantic network representation of the domain knowledge arose for several reasons as we began to work with Cohen and Axline, our collaborating experts. First, the knowledge of infectious disease therapy selection was ill-structured and, we found, difficult to represent using labeled arcs between nodes. Unlike South American geography, our domain did not have a clear-cut hierarchical organization, and we found it challenging to transfer a page or two from a medical textbook into a network of sufficient richness for our purposes. Of particular importance was our need for a strong inferential mechanism that would allow our system to reason about complex relationships among diverse concepts; there was no precedent for inferences on a semantic net that went beyond the direct, labeled relationships between nodes.[1]

Perhaps the greatest problem with a network representation, and the greatest appeal of production rules, was our gradually recognized need to deal with small chunks of domain knowledge in interacting with our expert collaborators. Because they were not used to dissecting their clinical reasoning processes, it was totally useless to ask them to "tell us all that you know." However, by discussing specific difficult patients, and by encour-

[1]The PROSPECTOR system (Duda et al., 1978a; 1978b), which was developed shortly after MYCIN, uses a network of inferential relations—a so-called *inference net*—to combine a semantic network with inference rules.

aging our collaborators to justify their questions or decisions, those of us who were not expert in the field began to tease out "nuggets" of domain knowledge—individual inferential facts that the experts identified as pertinent for problem solving in the domain. By encoding these facts as individual production rules, rather than attempting to decompose them into nodes and links in a semantic network, we found that the experts were able to examine and critique the rules without difficulty. This transparency of the knowledge base, coupled with the inherent modularity of knowledge expressed as rules, allowed us to build a prototype system quickly and allowed the experts to identify sources of performance problems with relative ease. They particularly appreciated having the ability to observe the effects of chained reasoning based on individual rules that they themselves had provided to us. In current AI terminology, the organization of knowledge was not object-centered but was centered around inferential *processes*.

Our early prototype rapidly diverged from DENDRAL because we were driven by different performance goals and different characteristics of the knowledge in the domain. Of particular importance was the need to deal with inexact inference; unlike the categorical conclusions in DENDRAL's rules, the actions in MYCIN's productions were typically conclusions about the state of the world that were not known with certainty. We soon recognized the need to accumulate evidence regarding alternative hypotheses as multiple rules lent credence to the conclusions. The need for a system to measure the weight of evidence of competing hypotheses was not surprising; it had also characterized conventional statistical approaches to computer-based medical decision making. Our certainty factor model, to which we refer frequently throughout this book (and which is the subject of Part Four), was developed in response to our desire to deal with uncertainty while attempting to keep knowledge modular and in rules.

The absence of complete certainty in most of our rules meant that we needed a control structure that would consider *all* rules regarding a given hypothesis and not stop after the first one had succeeded. This need for exhaustive search was distinctly different from control in DENDRAL, where the hierarchical ordering of rules was particularly important for correct prediction and interpretation (see Chapter 2). Because rule ordering was not important in MYCIN, the modularity of rules was heightened; the experts did not need to worry about ordering the rules they gave us or about other details of control.[2]

Another important distinction between the reasoning paradigms of DENDRAL and MYCIN was recognized early. DENDRAL generated hypotheses regarding plausible chemical structures and used its rule set to

[2]The arbitrary order of MYCIN's rules did lead to some suboptimal performance characteristics, however. In particular, the ordering of questions to the user often seemed unfocused. It was for this reason that the MAINPROPS (later known as INITIALDATA) feature was devised (see Chapter 5), and the concept of meta-rules was developed to allow rule selection and ordering based on strategic knowledge of the domain (see Chapter 28). The development of *prototypes* in CENTAUR (Chapter 23) was similarly motivated.

test these hypotheses and to select the best ones. Thus DENDRAL's control scheme involved forward invocation of rules for the last phase of the plan-generate-and-test paradigm. On the other hand, it was unrealistic for MYCIN to start by generating hypotheses regarding likely organisms or combinations of pathogens; there were no reasonable heuristics for pruning the search space, and there was no single piece of orienting information similar to the mass spectrum, which provided the planning information to constrain DENDRAL's hypothesis generator. Thus MYCIN was dependent on a reasoning model based on evidence gathering, and its rules were used to guide the process of input data collection. Because we wanted to avoid problems of natural language understanding, and also did not want to teach our physician users a specialized input language, we felt it was unreasonable to ask the physician to enter some subset of the relevant patient descriptors and then to have the rules fire in a data-driven fashion. Instead, we chose a goal-directed control structure that allowed MYCIN to ask the relevant questions and therefore permitted the physician to respond, in general, with simple one-word answers. Thus domain characteristics led to forward-directed use of the generate-and-test paradigm in DENDRAL and to goal-directed use of the evidence-gathering paradigm in MYCIN.

We were not entirely successful in putting all of the requisite medical knowledge into rules. Chapter 5 describes the problems encountered in trying to represent MYCIN's therapy selection algorithm as rules. Because therapy selection was initially implemented as LISP code rather than in rules, MYCIN's explanation system was at that time unable to justify specific therapy decisions in the same way it justified its diagnostic decisions. This situation reflects the inherent tension between procedural and production-based representation of this kind of algorithmic knowledge. The need for further work on the problem was clear. A few years later Clancey assumed the challenge of rewriting the therapy selection part of MYCIN so that appropriate explanations could be generated for the user. We were unable to encode the entire algorithm in rules, however, and instead settled on a solution reminiscent of the generate-and-test approach used in DENDRAL: rules were used to evaluate therapeutic hypotheses after they had been proposed (generated) by an algorithm that was designed to support explanations of its operation. This clever solution, described in Chapter 6, seemed to provide an optimal mix of procedural and rule-based knowledge.

3.1 Design Considerations

Many of the decisions that led to MYCIN's initial design resulted from a pragmatic response to perceived demands of physicians as computer users. Our perceptions were largely based on our own intuitions and observations about problems that had limited the success of previous computer-based medical decision-making systems. More recently we have undertaken formal studies of physician attitudes (Chapter 34), and the data that resulted, coupled with our prior experience building MYCIN, have had a major impact on our more recent work with ONCOCIN (Chapter 35). These issues are addressed in detail in Part Eleven.

However, since many of the features and technical decisions that are reflected in the other chapters in Part Two are based on our early analysis of design considerations for MYCIN (Shortliffe, 1976), we summarize those briefly here. We have already alluded to several ways in which MYCIN departed from the pure production systems described in Chapter 2. These are further discussed throughout the book (see especially Chapter 36), but it is important to recognize that the system's development was evolutionary. Most such departures resulted from characteristics of the medical domain, from our perceptions of physicians as potential computer users, or from unanticipated problems that arose as MYCIN grew in size and complexity.

We recognized at the outset that educational programs designed for instruction of medical students had tended to meet with more long-term success than had clinical consultation programs. A possible explanation, we felt, was that instructional programs dealt only with hypothetical patients in an effort to teach diagnostic or therapeutic concepts, whereas consultation systems were intended to assist physicians with the management of real patients in the clinical setting. A program aiding decisions that can directly affect patient well-being must fulfill certain responsibilities to physicians if they are to accept the computer and make use of its knowledge. For example, we observed that physicians had tended to reject computer programs designed as decision-making aids unless they were accessible, easy to use, forgiving of simple typing errors, reliable, and fast enough to save time. Physicians also seemed to prefer that a program function as a *tool*, not as an "all-knowing" machine that analyzes data and then states its conclusions as dogma without justifying them. We had also observed that physicians are most apt to need advice from consultation programs when an unusual diagnostic or therapeutic problem has arisen, which is often the circumstance when a patient is acutely ill. Time is an important consideration in such cases, and a physician will probably be unwilling to experiment with an "unpolished" prototype. In fact, time will *always* be an important consideration given the typical daily schedule of a practicing physician.

With considerations such as these in mind from the start, we defined the following list of prerequisites for the acceptance of a clinical consultation program (Shortliffe et al., 1974):[3]

[3]This analysis was later updated, expanded, and analyzed after we gained more experience with MYCIN (Shortliffe, 1980).

1. The program should be *useful*; i.e., it should respond to a well-documented clinical need and, ideally, should tackle a problem with which physicians have explicitly requested assistance.
2. The program should be *usable*; i.e., it should be fast, accessible, easy to learn, and simple for a novice computer user.
3. The program should be *educational when appropriate*; i.e., it should allow physicians to access its knowledge base and must be capable of conveying pertinent information in a form that they can understand and from which they can learn.
4. The program should be able to *explain its advice*; i.e., it should provide the user with enough information about its reasoning so that he or she can decide whether to follow the recommendation.
5. The program should be able to *respond to simple questions*; i.e., it should be possible for the physician to request justifications of specific inferences by posing questions, ideally using natural language.
6. The program should be able to *learn new knowledge*; i.e., it should be possible to tell it new facts and have them easily and automatically incorporated for future use, or it should be able to learn from experience as it is used on large numbers of cases.
7. The program's knowledge should be *easily modified*; i.e, adding new knowledge or correcting errors in new knowledge should be straightforward, ideally accomplished without having to make explicit changes to the program (code) itself.

This list of design considerations played a major role in guiding our early work on MYCIN, and, as we suggested earlier in this chapter, they largely account for our decision to implement MYCIN as a rule-based system. In Chapters 4 through 6, and in subsequent discussions of knowledge acquisition (Part Three) and explanation (Part Six), it will become clear how the production system formalism provided a powerful foundation for an evolving system intended to satisfy the design goals we have outlined here.

3.2 MYCIN as an Evolutionary System

One of the lessons of the MYCIN research has been the way in which the pure theory of production systems, as described in Chapter 2, has required adaptation in response to issues that arose during system development. Many of these deviations from a pure production system approach with backward chaining will become clear in the ensuing chapters. For reference we summarize here some of those deviations, citing the reasons for changes that were introduced, even though this anticipates more complete discussions in later chapters.

1. *The context tree*: We realized the need to allow our rules to make conclusions about multiple objects and to keep track of the hierarchical relationships among them. The context tree (described in Chapter 5) was created to provide a mechanism for representing hierarchical relationships and for quantifying over multiple objects. For instance, ORGANISM-1 and ORGANISM-2 are contexts of the same type that are related to cultures in which they are observed to be growing and that need to be compared, collected, and reasoned with together at times.

2. *Instantiation of contexts*: When a new object required attention, we needed a mechanism for creating it, naming it, and recording its associations with other contexts in the system. Prototypical contexts, similar in concept to the "frames" of more recent AI work (Minsky, 1975), provided a mechanism for creating new objects when they were needed. These are called *context-types* to distinguish them from individual contexts. For instance, ORGANISM is a context-type.

3. *Development of MAINPROPS*: Physicians using the evolving system began to complain that MYCIN did not ask questions in the order they were used to. For example, they indicated it was standard practice to discuss the site, timing, and method of collection for a culture as soon as it was first mentioned. Thus we created a set of parameters called the MAINPROPS for each prototypical context.[4] The values of these parameters were automatically asked for when a context was first created, thereby providing the kind of focused questioning with which physicians felt most comfortable. The benefit was in creating a more natural sequence of questions. The risk was in asking a few more questions than might be logically necessary for some cases. This was a departure from the pure production system aproach of asking questions only when the information was needed for evaluating the premise of a rule.

4. *Addition of antecedent rules*: The development of MAINPROPS meant that we knew there were a small number of questions that would be asked every time a context was created. In a pure backward-chaining system, rules that had premise conditions that depended *only* on the values of parameters on MAINPROPS lists would be invoked when needed so there was no *a priori* reason to do anything special with such rules. However, two situations arose that made us flag such rules as antecedent rules to be invoked in a data-driven fashion rather than await goal-oriented invocation. First, there were cases in which an answer to one MAINPROPS

[4]This name was later changed to INITIALDATA in EMYCIN systems.

question could uniquely determine (via a definitional antecedent rule) the value of another subsequent MAINPROPS property for the same context (e.g., if an organism's identity was known, its gram stain and morphology were of course immediately determined). By implementing such rules as antecedent rules and by checking to see if the value of a MAINPROPS parameter was known before asking the user, we avoided inappropriate or unnecessary questions.

The second use of antecedent rules arose when the preview mechanism was implemented (see paragraph 12 below). Because an antecedent rule could determine that a premise condition of another rule was false, such rules could be rejected immediately during the preview phase. If antecedent rules had been saved for backward-chained invocation, however, the preview mechanism would have failed to reject the rule in question. Thus the MONITOR would have inappropriately pursued the first two or three conditions in the premise of the rule, perhaps at considerable computational expense, only to discover that the subsequent clause was clearly false due to an answer of an earlier MAINPROPS question. Thus antecedent rules offered a considerable enhancement to efficiency in such cases.

5. *Self-referencing rules*: As will be discussed in Chapter 5, it became necessary to write rules in which the same parameter appeared in both the premise and the action parts. Self-referencing rules of the form A & B & C $\rightarrow$ A are a departure from the pure production system approach, and they required changes to the goal-oriented rule invocation mechanism. They were introduced for three purposes: default reasoning, screening, and using information about risks and utilities.

a. *Default reasoning*: MYCIN makes no inferences except those that are explicitly stated in rules, as executed under the certainty factor (CF) model (see Chapter 11) and backward-chaining control. There are no implicit ELSE clauses in the rules that assign default values to parameters.[5] When rules fail to establish a value for a parameter, its value is considered to be UNKNOWN—no other defaults are used. One use of the self-referencing rules is to assign a default value to a parameter explicitly:

> IF a value for X is not known (after trying to establish one),
> THEN conclude that the value of X is Z.

Thus, reasoning with defaults is done in the rules and can be explained in the same way as any other conclusions. The control structure had to be changed, however, to delay executing these rules until all other relevant rules had been tried.

b. *Screening*: For purposes of human engineering, we needed a screen-

[5]Explicit else clauses were defined in the syntax (see Chapter 5) but were eliminated, mostly for the sake of simplicity.

ing mechanism to avoid asking about unusual parameters (B and C, above) unless there is already some *other* evidence for the hypothesis (A) under consideration. For example, we did not want MYCIN to use the simple rule

Pseudomonas-type skin lesions $\rightarrow$ *Pseudomonas*

unless there already was evidence for *Pseudomonas*—otherwise, the program would appear to be asking for minute pieces of data inappropriately.

c. *Utilities*: Self-referencing rules gave us a way to consider the risks of failing to consider a hypothesis. Once there is evidence for *Pseudomonas*, say, being a possible cause of an infection, then a self-referencing rule can boost the importance of considering it in therapy, based on the high risk of failing to treat for it.

6. *Mapping rules*: We soon recognized the need for rules that could be applied iteratively to a set of contexts (e.g., a rule comparing a current organism to each bacterium in the set of all previous organisms in the context tree). Special predicate functions (e.g., THERE-IS, FOR-EACH, ONE-OF) were therefore written so that a condition in a rule premise could map iteratively over a set of contexts. This was a partial solution to the general representation problem of expressing universal and existential quantification. Only by considering all contexts of a type could we determine if all or some of them had specified properties. The context tree allowed easy comparisons within any parent context (e.g., all the organisms growing in CULTURE-2) but did not allow easy comparison across contexts (e.g., all organisms growing in all cultures).

7. *Tabular representation of knowledge*: When large numbers of rules had been written, each having essentially the same form, we recognized the efficiency of collapsing them into a single rule that read the values for its premise conditions and action from a specialized table. (A related concept was implemented in changes that allowed physicians to enter information in a more natural way. If they were looking at a patient's record for answers to questions, it was more convenient to enter many items at once into a table of related parameters. There was, however, the attendant risk of asking for information that would not actually be used in some cases.) Chapter 5 describes the implementation of this feature.

8. *Augmentation of rules*: As multiple experts joined to collaborate on development of the knowledge base, we recognized the need to keep track of who wrote individual rules. Thus extra properties were added to rules that allowed us to keep track of authorship, to record literature references that defended the inference stored in the rule, and to allow recording of free-form text justification of certain complicated rules for which the normal rule translation was somewhat cryptic. These extra *slots* associated with

rules gave the latter more the character of frames than of pure productions.

9. *The therapy algorithm*: As described in Chapter 5, the final step in MYCIN's decision process was largely algorithmic and proved difficult to encode in rules. Chapter 6 describes our eventual solution, in which we integrated algorithmic and rule-based approaches in a novel manner.

10. *Management of uncertainty*: Previous PS's had not encoded the uncertainty in rules. Thus MYCIN's certainty factor model (see Part Four) was an augmentation mandated by the nature of decision making in this complex medical domain.

11. *Addition of meta-rules*: As mentioned in Chapter 2 and described in Chapter 28, we began to realize that strategies for optimal rule invocation could themselves be encoded in rules. MYCIN's PS approach was modified to manage high-level meta-rules that could be invoked via the usual rule monitor and that would assist in determining optimal problem-solving strategies.

12. *Addition of a preview mechanism*: It became clear that it was inefficient for the rule interpreter to assess the first few conditions in a rule premise if it was already known that a subsequent condition was false. Thus a preview mechanism was added to the interpreter so that it first examined the whole premise to see if there were parameters whose values had previously been determined. The addition of the preview mechanism made it important to add antecedent rules, as mentioned above (paragraph 4).

13. *The concept of a unity path*: Because many MYCIN rules reached conclusions with less than certainty, it was generally necessary to invoke *all* rules that could bear on the value of a parameter under consideration. This is part of MYCIN's cautious evidence-gathering strategy in which *all* relevant evidence available at the time of a consultation is used. However, if a rule successfully reaches a conclusion with certainty (i.e., it has CF = 1), then it is not necessary to try alternate rules. Thus the rule monitor was altered to try first those rules that could reach a conclusion with certainty, either through a single rule with CF = 1 or through a chain of rules, each with CF = 1 (a so-called unity path). When certain rules succeeded, the alternate rules were ignored, and this prevented inefficiencies in the development of the reasoning network and in the generation of questions to the user.

14. *Prevention of circular reasoning*: The issue of circular reasoning does not normally arise in pure production systems but was a serious potential problem for MYCIN. (Self-referencing rules, discussed in paragraph 5 above, are a special case of the general circular reasoning problem

involving any number of rules.) Special changes to the rule monitor were required to prevent this undesirable occurrence (see Chapter 5).

15. *The tracing mechanism*: As is described in Chapter 5, we made the decision to determine *all* possible values of a parameter instead of determining only the value specified in the premise condition of interest. This potential inefficiency was tolerated for reasons of user acceptance. We found that physicians preferred a focused and exhaustive consideration of one topic at a time, rather than having the system return subsequently to the subject when another possible value of the same parameter was under consideration.

16. *The ASKFIRST concept*: Pure production systems have not generally distinguished between attributes that the user may already know with certainty (such as values of laboratory tests) and those that inherently require inference. In MYCIN this became an important distinction, which required that each parameter be labeled as an ASKFIRST attribute (originally named LABDATA as discussed in Chapter 5) or as a parameter that should first be determined by using rules rather than by asking the user.

17. *Procedural conditions associated with parameters*: We also discovered unusual circumstances in which a special test was necessary before MYCIN could decide whether it was appropriate to ask the user for the value of a parameter. This was solved through a kind of procedural attachment, i.e., an executable piece of conditional code associated with a parameter, which would allow the rule monitor to decide whether a question to the user was appropriate. Each parameter thus began to be represented as a frame with several slots, including some whose values were procedures.

18. *Rephrasing prompts*: As users became more familiar with MYCIN, we found that they preferred short, less detailed prompts when the program requested information. Thus a "terse" mode was implemented and could be selected by an experienced user. Similarly, a reprompt mechanism was developed so that a novice user, puzzled by a question, could be given a more detailed explanation of what MYCIN needed to know. These features were added to an already existing HELP facility, which showed examples of acceptable answers to questions.

19. *Multiple instances of contexts*: Some of the questions asked by MYCIN are necessary for deciding whether or not to create contexts (rather than for determining the value of a parameter). Furthermore, optimal human engineering requires that this kind of question be phrased differently for the first instance of a context-type than for subsequent instances. These alternate prompts are discussed in Chapter 5.

20. *HERSTORY List*: Another addition to the rule monitor in MYCIN was a mechanism for keeping track of all rules invoked, failing, succeeding, etc., and the reasons for these various outcomes. The so-called HERSTORY List, or history tree, then provided the basis for MYCIN's explanations in response to users' queries.

21. *Creation of a Patient Data Table*: Finally, we recognized the need to develop mechanisms for (a) reevaluating cases when more information became available and (b) assessing the impact of modifications to the knowledge base on a library of cases previously handled well. These goals were achieved by the development of a Patient Data Table, i.e., a mechanism for storing and accessing the initializing conditions necessary for full consideration of cases. See Chapter 5 for further discussions of this feature.

3.3 A Word About the Logic of MYCIN

The logic of MYCIN's reasoning is propositional logic, where the elementary propositions are fact triples and the primary rule of inference is *modus ponens* (A and A $\supset$ B implies B). It is extended (and somewhat complicated) in the following respects:

- Certainty factors (CF's) are attached (or propagated) to all propositions.
- CF's are associated with all implications.
- Predicates are associated with fact triples to change the way facts stated in rules are matched against facts in the dynamically constructed case record. A variety of predicates have been defined (see Section 5.1.5); some refer to values of attributes (e.g., NOT-SAME, ONE-OF) and some reference values of CF's (e.g., KNOWN, DEFINITE).
- Limited quantification is allowed over conjunctions of propositions (e.g., THERE-IS, FOR-EACH).
- Meta-level reasoning is allowed in order to increase efficiency (e.g., using meta-rules or looking for a unity path).

MYCIN's logic is incomplete in the sense that we know there are propositions that can be expressed in the language but are not provable as theorems. MYCIN's logic is not inconsistent in itself (we believe), but it is *not* immune to inconsistencies introduced into its knowledge base.

3.4 Overview of Part Two

The remainder of this part consists of three papers that summarize MYCIN and its use of production rules. In order to orient the reader to MYCIN's overall motivation and design, we first include as Chapter 4 an introductory paper that provides an overview of the system as of 1978 (approximately the time when development of the medical knowledge base stopped). Chapter 5 is the original detailed description of MYCIN from 1975. It provides technical information on the system's representation and control mechanisms. Chapter 6 is a brief paper from 1977 that discusses the way in which production rules were adapted to deal with the algorithmic knowledge regarding therapy selection.