August 1982

Report No. STAN-CS-82-926

Also numbered: HPP-82-14

Principles of Rule-Based Expert Systems

by

Bruce G. Buchanan and Richard O. Duda

Department of Computer Science

Stanford University
Stanford, CA 94305

Heuristic Programming Project
Report No. HPP-82-14

also:
Fairchild
Technical Report No.626

August 1982

Principles of Rule-Based Expert Systems

BRUCE G. BUCHANAN
HEURISTIC PROGRAMMING PROJECT
DEPARTMENT OF COMPUTER SCIENCE
STANFORD UNIVERSITY

RICHARD O. DUDA
LABORATORY FOR ARTIFICIAL INTELLIGENCE RESEARCH
FAIRCHILD CAMERA AND INSTRUMENT CORPORATION
PALO ALTO, CALIFORNIA

.The Stanford University component of this research is funded in part by ARPA contract # MDA903-80-C-0107, NIH contract # NIH RR 0078509, ONR contract # NO001 4-79-C-0302, and Schlumberger-Doll Research Laboratory.

To appear *in M. Yovits (ed.)* **Advances in Computers, Vol.22,**
New York: Academic Press

Table of Contents

Principles of Rule-Based Expert Systems .

Bruce G. Buchanan
Richard 0. Duda

1 INTRODUCTION: WHAT IS AN EXPERT SYSTEM?

An expert system is a computer program that provides expert-level solutions to 'important problems and is:

- heuristic -- i.e., it reasons with judgmental knowledge as well as with formal knowledge of established theories;
- transparent -- i.e., it provides explanations of its line of reasoning and answers to queries about its knowledge;
- flexible -- i.e., it integrates new knowledge incrementally into its existing store of knowledge.[1].

The key ideas have been developed within Artificial Intelligence (AI) over the last fifteen years, but in the last few years more and more applications of these ideas have been made. The purpose of this article is to familiarize readers with the architecture and construction of one important class of expert systems, called rule-*based systems.* In this overview, many programs and issues are necessarily omitted, but we attempt to provide a framework for understanding this advancing frontier of computer science.

Because computers are general symbol manipulating devices, the non-numeric and heuristic aspects of problem solving can be encoded in computer programs as well as the mathematical and algorithmic aspects [Newell76]. Artificial Intelligence research has focused on just this point. Work on expert systems is, in one sense, the applied side of AI, in which current techniques arc applied to problems to provide expert-level help on those problems. However, there is more to building an expert system than straightforward application of AI techniques. In this early stage of dcvelopmcnt, each new application challenges the current stock of ideas, and many applications force extensions and modifications.

We focus on rule-based systems in this survey because they clearly demonstrate the state of the-art in building expert systems and illustrate the main issues. In a rule-based system, much of the knowledge is represented as rules, that is, as conditional sentences relating statements of facts with one another. *Modus ponens* is the primary rule of inference by which a system adds new facts to a growing data base:

---

[1]We use the term 'expert system' more restrictively than many authors. In doing so, wc distinguish expert systems from, among others, programs that perform well but cannot bc examined and programs that arc examinable but do not perform well and programs that solve problems for which no special expertise is required. For other recent reviews of the state of the art of expert systems, see [Bonnet81, Buchanan81, Davis82a, Duda81, Feigenbaum79, Hayes-Roth78, Infotech81, Michie79, Michie80, Pinson81, Stefik82, Waterman78]

```
IF B IS TRUE                          B
AND B IMPLIES C,          OR          B --> c
THEN C IS TRUE.                       --------
                                      C
```

Conceptually, the basic framework of a rule-based. system is simple; the variations needed to deal with the complexities of real-world problems make the framework interestingly more complex. For example, the rule B-->C is often interpreted to mean "B suggests C", and strict deductive reasoning with rules gives way to plausible reasoning. Other methodologies are also mentioned and briefly discussed.

In an expert system the fundamental assumption is *"Knowledge is Power".* Specific knowledge of the task is coupled with general problem solving knowledge to provide expert-level analyses of difficult situations. For example, MYCIN (described below) analyzes medical data about a patient with a severe infection, PROSPECTOR [Duda79] analyzes geological data to aid in mineral exploration, and PUFF [Kunz78] analyzes the medical condition of a person with respiratory problems. In order to provide such analyses, these systems need very specific rules containing the necessary textbook and judgmental knowledge about their domains.

The key idea is to separate knowledge of the task area as much as possible from the procedures that manipulate it. This promotes flexibility and transparency because the store of knowledge -- called the *knowledge base* -- can then be manipulated and examined as any other data structures. Separation does not guarantee flexibility or transparency, nor does it guarantee high performance. But if a packaged framework of inferences and control procedures can be used, the design of a new system properly focuses on the *expertise* needed for high performance.

For many years, AI research has focused on heuristic reasoning. Heuristics, or rules of thumb, arc an essential key to intelligent problem solving because computationally feasible, mathematically precise methods arc known for only a relatively few classes of problems. A large part of what an expert system needs to know is the body of heuristics that specialists use in solving hard problems. Specialists in science, mathematics, medicine or any discipline do not confine their everyday reasoning to the axiomatic, formal style of stereotyped textbook accounts. An expert system can also benefit from reasoning with informal knowledge. There is no explicit attempt to *simulate* a specialist's problem solving behavior in an expert system. However, the system derives power from integrating the same heuristic knowledge as experts use with the same informal style of reasoning.

The first expert systems, DENDRAL [Lindsay80] and MACSYMA [Moses71], emphasized performance, the former in organic chemistry and the latter in symbolic integration. These systems were built in the mid-1960's, and were nearly unique in AI because of their focus on real-world problems and on specialized knowledge. In the 1970's, work on expert systems began to flower, especially in medical problem areas (see, for example [Pople77, Shortliffe76, Szolovits78, Weiss79b]). The issues of making the system understandable through explanations [Scott77, Swartout81] and of making the system flexible enough to acquire new knowledge [Davis79, Mitchell791 were emphasized in these and later systems.

In the early work, the process of constructing each new system was tedious because each was custom-crafted. The major difficulty was acquiring the requisite knowledge from experts and reworking it in a form fit for machine consumption, a process' that has come to be known as *knowledge engineering.* One of the most important developments of the late 1970's and early 1980's is the construction of several *knowledge engineering frameworks* designed to aid in building, debugging, interpreting, and explaining expert systems. Engineering an expert's knowledge into a usable form for a program is a formidable task. Thus computer-based aids for system builders are important. Current tools -- including EMYCIN [vanMelle80] ROSIE [Fain81], KAS [Reboh81], EXPERT [Weiss79a], and OPS [Forgy77] -- provide considerable help. For example, working prototypes of new expert systems with one or two dozen rules have been constructed in a few days using EMYCIN [Bennett81a]. Considerable effort is then required to refine the knowledge base, but the effort is focused on the knowledge and not on the code. Much research remains, however, before broad, powerful systems can be constructed quickly.

1.1 Example: The MYCIN Program

MYCIN is a rule-based system developed in the mid-late 1970's at Stanford University. Its representation and architecture are described in detail in [Davis77b] and [Shortliffc, 1976]. Although it is now several years old, it is representative of the state of the art of expert systems in its external behavior, which is shown in the following excerpt from a dialogue between the MYCIN program and a physician. It illustrates the interactive nature of most rule-based systems and provides a single example for the rest of the discussion.

The task of MYCIN is to help diagnose the likely causes of a patient's infection and to help recommend the best therapy for that individual patient. The consultation is driven by a knowledge base of about 450 rules and a thousand facts about medicine, mostly about meningitis infections. In the example, we see the system asking questions to obtain a description of a new case, and we see the system providing an explanation of its line of reasoning.

The main point of this example is that a simple rcprcscntation of knowledge as rules, and a relatively simple control structure, are adequate for constructing a consultation system in a domain as complex (but narrow!) as meningitis diagnosis and therapy. Appendix A shows answers to user's questions about this consultation and about the knowledge base.

Edited Transcript of a MYCIN Consultation Session

[User's responses are in upper case and follow double asterisks]

[Editorial comments are in italics]

```
--------PATIENT-538--------
1) Patient's name:
** PT538
2) Age:
** 34 YEARS
3) Sex:
** MAEL
```

MYCIN respells the user's answer

```
=MALE
4) Race:
** CAUCASIAN
```

Several questions are skipped.

```
14) Has Pt.538 recently had symptoms of persistent headache or other
    abnormal neurologic symptoms (dizziness, lethargy, etc.)?
** YES
15) Has Pt538 recently had objective evidence of abnormal neurologic
    signs (nuchal rigidity, coma, seizures, etc.) documented by physician
    observation or examination?
** YES
```

Note that MYCIN has concluded and informed the user that there is a likely meningitis infection and pursues this line of inquiry.

```
The CSF cultures will be considered to be associated with MENINGITIS.

16) Please give the date on which clinical evidence (symptoms, signs, or
    laboratory tests) of the meningitis first appeared.
** 29-Jan-77 22:15

.

23) For how many days has Pt538 had abnormal neurologic signs?
** 7.2 HOURS
```

Note that MYCIN accepts the different units of time

```
28) Do you know the results of a recent CBC of Pt538?
** YES

Please enter CBC results in the following table
```

```
       WBC                        Percent      Percent|
       in 1000's                  PMN          BANDS
29)** 1.9                         UNKNOWN      UNKNOWN
```

Note that MYCIN accepts "UNKNOWN" as a legitimate value. It does not need complete, or even correct, data for all answers.

```
30) Has Pt538 had any injury or insult to, or defect in the central- .
    nervous-system (excluding neurosurgery)?
** NO



35) Is meningitis a hospital-acquired infection?
** WHY
```

The user asks "WHY" to see an explanation of the line of reasoning MYCIN is following. The program first rephrases the question and then shows the rule whose premise it is trying to establish.

```
[i.e. WHY is it important to determine whether the infection was acquired
    while the patient was hospitalized?]

[2.0] This will aid in determining the organisms (other than those seen on
    cultures or smears) which might be causing the infection.
    It has already been established that
       [2.1] the infection which requires' therapy is meningitis, and
       [2.2] organisms were not seen on the stain of this pending csf
                culture, and
       [2.3] the type of the infection is bacterial, and
       [2.4] neurosurgery is not one of the relevant items from the history
                of pt538, and
       [2.5] the age of pt538 is greater than 10 days
    Therefore, if
       [2.6] the infection was acquired while the patient was hospitalized
    then
       there is evidence that the organisms (other than those seen on
          cultures or smears) which might be causing the infection is
          e.coli (.75) staphylococcus-coag-pos (.3) pseudomonas-aeruginosa
          (.3) klebsiella-pneumoniae (.5)
  Author: Yu
    [RULE545]
```

The numbers in parentheses following each conclusion indicate the measure of importance of the conclusion. This single number combines frequency and utility considerations, as discussed in section 4.

[back to question 35...]
**** HOW 2.3**

At this point the user is asking for additionaf clarification of how clause 2.3 of the above rule has already been determined. MYCIN shows the numbers of the rules used to make conclusion and will print any of them upon request.

[i.e. HOW was it established that the type of the infection is bacterial?]

The following rulesconcluded about:
the likelihood that the type of the infection is bacterial

	Rule	Yes	No
[2.7]	RULE148	(.19)	
[2.8]	RULE500	(.51)	
[2.9]	RULE501	[illegible]	
[3.0]	RULE502	[illegible]	
[3.1]	RULE526	[illegible]	
[3.2]	RULE504	(.97)	
[3.3]	RULE524	(.98)	

Thus, it has been established that the type of the infection is bacterial.

MYCIN now prints intermediate conclusions to show the user what it it is considering the likely causes of the problem to be.
Further questions are motivated by these conclusions.

I have found evidence (RULE545 RULE557) that treatment should cover for E.coli causing meningitis.
I have found evidence (RULE545 RULE557 RULE578) that treatment should cover for Pseudomonas-aeruginosa causing meningitis.
I have found evidence (RULE545 RULE557) that treatment should cover for Klebsiella-pneumoniae causing meningitis.
I have found evidence (RULE507) that treatment should cover for Diplococcus-pneumoniae causing meningitis.

40) Does Pt538 have focal neurological signs?
**** UNKNOWN**

.
.
.

MYCIN now prints the intermediate conclusions about the possible causes of the problem that are most important to consider in constructing a rational therapy plan,

```
   My therapy recommendations will be designed to treat for organisms
that either are very likely or, although less likely, would have a
significant effect on therapy selection if they were present. It is
important to cover for the following probable infection(s) and
associated organism(s):


INFECTION-l is MENINGITIS

  + <ITEM-l> E.COLI [from clinical evidence only]
  + <ITEM-Z> PSEUDOMONAS-AERUGINOSA [from clinical evidence only]
  + <ITEM-3> KLEBSIELLA-PNEUMONIAE [from clinical evidence only]
  + <ITEM-4> DIPLOCOCCUS-PNEUMONIAE [from clinical evidence only]

   41) Does Pt538 have a clinically significant allergic reaction to any
       antimicrobial agent?
   ** NO
[Considering AMPICILLIN-AND-GENTAMICIN for use against INFECTION-l...]
   42) Do you have reason to suspect that Pt538 may have impaired renal
       function?
   ** NO

   43) Pt538's weight in kilograms (or <number> POUNDS):
   ** 70
```

The final conclusion is about the best therapy to cover for the likely causes of the infection, as determined from the incomplete evidence available.

```
[REC-1] My preferred therapy recommendation is as follows:
   In order to cover for Items <1 2 3 4>:
        Give the following in combination:
        1) AMPICILLIN
        Dose: 3.5g (28.0 ml) q4h IV [calculated on basis of 50 mg/kg] . .

        2) GENTAMICIN
        Dose: 119 mg (3.0 ml, 80mg/2ml ampule) q8h IV [calculated on basis
              of 1.7 mg/kg] plus consider giving 5 mg q24h Intrathecal
        Comments: Monitor serum concentrations


        Since high concentrations of penicillins can inactivate
        aminoglycosides, do not mix these two antibiotics in the same IV
        bottle.


Do you wish to see the next choice therapy?
** NO
```

The user may have reasons to prefer another combination of drugs. At this point, MYCIN is prepared to show other therapies, compare its preferred therapy to one the user specifies or to *answer questions about any part of the consultation or about the items in its knowledge base. Appendix A shows some of these explanations.*

The level of expertise MYCIN exhibited on meningitis test cases in a double-blind evaluation study equalled the level of expertise of faculty members in Infectious Diseases [Yu79]. However, the program was never put into routine use in hospitals. Factors other than the program's competence, such as human engineering and exportability, were the main barriers to routine use. There are two ways of overcoming these kinds of problems. First, some follow-on research to MYCIN addresses the human engineering problems directly, for example, by integrating high quality graphics with user-oriented forms and charts for input and output [Shortliffe81]. Second, some MYCIN-like programs finesse many human engineering problems by collecting data from on-line instruments rather than from users [Kunz78]. Exportability can be gained by rewriting [Carhart79, Kunz78] or by designing for export initially [Weiss79a].

1.2 Key Components

The example showed some of the characteristic features of an expert system-- the heuristic nature of MYCIN'S rules, an explanation of its line of reasoning, and the modular form of rules in its knowledge base. We postponed a discussion of the general structure of a system until after the example, and defer entirely specific questions about implementation. For discussing the general structure, we describe a generalization of MYCIN, called EMYCIN [vanMelle80] for "essential MYCIN". It is a framework for constructing and running rule-based systems, like MYCIN.

Generally speaking, an expert system requires a *knowledge base* and an *inference procedure*. The knowledge base for MYCIN is a set of rules and facts covering specialized knowledge of meningitis as well as some general knowledge about medicine. The inference procedure in MYCIN (and all systems constructed in EMYCIN) is a large set of INTERLISP functions that control the interaction and update the current state of knowledge about the case at hand. The sections below on Representation and Inference discuss these two main parts. A system also requires a global *database* of facts known or inferred about a specific case, the working dataset, in other words. And it requires an *interface* program that makes the output understandable to users and translates users' input into internal forms. MYCIN uses a technical subset of English in which there is little ambiguity in the language of communication with users.

In addition to these four parts, the EMYCIN system contains an *explanation subsystem* to answer questions about a consultation or about the static knowledge base. It also contains a *knowledge base editor* to aid in the construction of new knowledge bases (and thus new systems) and to aid in debugging an emerging knowledge base. All of these components are shown schematically in the figure below.

```
             ----------------------        ---------------
EXPERT'S    |Knowledge Base Editor|        |  Customized  |
RULES --->>I  and Debugging Aids  |--->>   |  Knowledge   |
            |                     |        |     Base     |
             ----------------------        ---------------
                                                  ↑
                                                  ↓
              ------------------          -----------------
             |   Interaction     |        | Problem Solver |
USER'S -->>I       Handler       |  --->> |                I-->> ADVICE FOR
DESCRIPTION|                     |        -----------------       THIS CASE
OF NEW CASE| .                   |                ↑
             |                   |                ↓
             |                   |        -----------------
             |                   |        |  Database of   |
             |                   |        |  Facts About   |
             |          and      |        |   This Case    |
             |                   |        -----------------
             |                   |                ↑
             |                   |                ↓
             |                   |        -----------------
             |                   |        |                |
USER'S -->>|Question-Answering  |        |  Explanation    |
QUESTIONS |       Routines     |  --->> |    Subsystem    I-->> EXPLANATIONS
             |                   |        |                |
              ------------------          -----------------
```

FIGURE 1. Parts of the EMYCIN system showing the separation of the knowledge base from problem solving procedures and other parts of the system.

We turn now to two basic issues that have become the central foci of work on expert systems: (a) how knowledge of a task area is represented in the computer program, and (b) how knowledge is used to provide expert-level solutions to problems.

2 REPRESENTATION OF KNOWLEDGE

A representation is a set of conventions for describing the world. In the parlance of AI, the representation of knowledge is the commitment to a vocabulary, data structures, and programs that allow knowledge of a domain to be acquired and used. This has long been a central research topic in AI (see [Amarel81, Barr81, Brachman80, Cohen82] for reviews of relevant work).

The results of 25 years of AI research on representation have been used to establish convenient ways of describing parts of the world. No one believes the current. representation methods are the final word.

However, they are well enough developed that they can be used for problem solving in interesting domains. As pointed out above, a central concern is separation of the choice of vocabulary and data structures from the choice of program logic and language. By separating a program's knowledge base from the inference procedures that work with the knowledge, we have attained some success in building systems that are understandable and extendable.

Three basic requirements on a representation scheme in an expert system 'are extendability, simplicity and explicitness.

Extendability -- the data structures and access programs must be flexible enough to allow extensions to the knowledge base without forcing substantial revisions. The knowledge base will contain heuristics that are built out of experts' experience. Not only do the experts fail to remember all relevant heuristics they use, but their experience gives them new heuristics and forces modifications to the old ones. New cases require new distinctions. Moreover, the most effective way we have found for building a knowledge base is by incremental improvement. Experts cannot define a complete knowledge base all at once for interesting problem areas, but they can define a subset and then refine it over many weeks or months of examining its consequences. All this argues for treating the knowledge base of an expert system as an open-ended set of facts and relations, and keeping the items of knowledge as modular as possible.

Simplicity -- We have all seen data structures that were so baroque as to be incomprehensible, and thus unchangeable. The flexibility wc argued for above requires conceptual simplicity and uniformity so that access routines can be written (and themselves modified occasionally as needed). Once the syntax of the knowledge base is fixed, the access routines can be fixed to a large extent. Knowledge acquisition, for example, can take place with the expert insulated from the data structures by access routines that make the knowledge base *appear* simple, whether it is or not. However, new reasons will appear for accessing the knowledge base as in explanation of the contents of the knowledge base, analysis of the links among items, display, or tutoring. With each of these reasons, simple data structures pay large benefits. From the designer's point of vi& there are two ways of maintaining conceptual simplicity: keeping the form of knowledge as homogeneous as possible or writing special access functions for non-uniform representations.

There is another sense of simplicity that needs mentioning as well. That is the simplicity that comes from using roughly the same terminology as the experts use. Programmers often find ingenious alternative ways of representing and coding what a specialist has requested, a fact that sometimes makes processing more "efficient" but makes modifying the knowledge base a nightmare.

Explicitness -- The point of representing much of an expert's knowledge is to give the system a rich enough knowledge base for high-performance problem solving. But because a knowledge base must be built incrementally, it is necessary to provide means for inspecting and debugging it easily. With items of knowledge represented explicitly, in relatively simple terms, the experts who are building knowledge bases can determine what items are present and (by inference) which are absent,

To achieve these goals, three types of representation framework have been used in expert systems. Although. we concentrate on rule-based systems in the sections below, we also mention frame-based and logic-based systems by way of contrast. Such frameworks are often called *representation languages* because, as with other programming languages, their conventions impose a rigid set of restrictions on how one can express and reason about facts in the world. In all of these languages, one can express conditional expressions and. causal dependencies. In rule-based systems, however, one sacrifices the ability to express many other general kinds of relations in favor of the homogeneity and simplicity of conditional rules. These frameworks, also, often include inference and control routines making them even more like languages.

2.1 Rule-Based Representation Frameworks

2.1 .1 Production Systems

Rule-based expert systems evolved from a more general class of computational models known as *production systems* [Newell73]. Instead of viewing computation as a prespecified sequence of operations, production systems view computation as the process of applying transformation rules in a sequence determined by the data. Where some rule-based systems [McDermott80] employ the production-system formalism very strictly, others such as MYCIN have taken great liberties with it.[2] However, the production system framework provides concepts that are of great use in understanding all rule-based systems.

A classical production system has three major components: (1) a *global database* that contains facts or assertions about the particular problem being solved, (2) a *rulebase* that contains the general knowledge about the problem domain, and (3) a *rule interpreter* that carries out the problem solving process.

The facts in the global database can be represented in any convenient formalism, such as arrays, strings of symbols, or list structures. The rules have the form

IF <condition> THEN <action> .

---

[2]The production system viewpoint has been employed in artificial intelligence work in two different ways -- as a model of human cognitive processes [Newell72] and as a framework for pattern-directed inference systems [Waterman78]. For a clear comparison of the different styles of use of production systems, see [Davis76].

In general, the left-hand-side (LHS) or condition part of a rule can be any pattern that can be matched against the database. It is usually allowed to contain variables that might be bound in different ways, depending upon how the match is made. Once a match is made, the right-hand-side (RHS) or action part of the rule can be executed. In general, the action can be any arbitrary procedure employing the bound variables. In particular, it can result in addition of new facts to the database, or modification of old facts in the database.

The rule interpreter has the task of deciding which rules to apply. It decides how the condition of a selected rule should be matched to the database, and monitors the problem-solving process. When it is used in an interactive program, it can turn to the user and ask for information (facts) that might permit the application of a rule.

The strategy used by the rule interpreter is called *the control strategy*. The rule interpreter for a classical production system executes rules in a "recognize-act" cycle. Here the rule interpreter cycles through the condition parts of the rules, looking for one that matches the current data base, and executing the associated actions for (some or all) rules that do match. As wc point out in Section 3, there are many other ways to control the application of rules, but in all cases the result of executing actions is to change the data base, enabling the application of some rules and disabling others.

At this level of generality, production systems are capable of arbitrarily complex behavior. The many ways in which conditions might be matched and variables might be bound, the many factors that might be important for rule selection, and the complicated effects of executing the rule actions can quickly lead to very difficult control problems. As one specific example, in many problem solving systems the application of one rule can invalidate conditions needed for the application of a previously applied rule; to cope with such possibilities, the rule interpreter may have to employ backtracking strategies, or may have to maintain and search detailed records of the interdependencies between facts in the database.

Many of the expert systems constructed to date are have controlled this complexity by sacrificing the ability to perform general problem-solving tasks. They have achieved their competence by specializing -- by exploiting the fallible but effective heuristic methods that human experts bring to a particular class of problems. Many of the high-performance systems (including MYCIN) can be characterized as simple *deduction systems,* programs in which a fact once entered in the global database (whether by the user or by the application of a rule) is never subsequently deleted. Their actions are typically limited to the addition of new facts to the database. In the remainder of this section, we use EMYCIN as a concrete, specific example of a rule-based approach to expert systems. While other rule-based systems have made other design tradeoffs, EMYCIN illustrates well the issues that arc involved.

2.1.2 EMYCIN Viewed as a Production System

To see how EMYCIN uses the production system formalism to represent knowledge, we must see how it represents facts about the current problem in its database, and how it represents general knowledge in its rules.

In all EMYCIN systems, facts are *associative triples,* that is, attribute-object-value triples, with an associated degree of certainty (called a *certainty factor,* or CF). More generally, the EMYCIN syntax for statements of fact (both within the database and within rules) is:

The <attribute> of <object> is <value> with certainty <CD .

In the MYCIN dialog shown above, fact triples are shown in the explanations as individual clauses of rules. For example, after Question #35, one fact that has been established is *"the type of the infection is bacterial."* It can, also be seen that each question is asking for a value to be associated with an attribute of an object.[3] *In* Question # 35, for example, MYCIN is asking whether or not *the infection of the patient is hospital-acquired.*

A rule is a conditional sentence relating several fact statements in a logical relation. The nature of the relation varies from rule to rule. Often rules record mere empirical associations -- rules of thumb based on past experience with little theoretical justification. Other rules are statements of theoretical associations, definitions, or causal laws.

The condition part of an EMYCIN rule is called its *premise.* In general, an EMYCIN premise is the conjunction of a number of *clauses.* Each clause is a simple statement concerning the facts, such as *"the age of the patient is greater than 10 days,"* or *"the identity of the infection is unknown."* To enable EMYCIN to query the database to determine to what degree a clause might be true, each clause includes a *matching predicate* **that** specifies how the statement is to be compared against the database of facts. In particular, a matching predicate is. not required to return a binary answer, but may return a number between 0 and 1 indicating how much evidence there is that the predicate is satisfied. About two dozen matching predicates are a standard part of EMYCIN, including *is the same as, is not the same as, has already been established, is not known, is greater than* (for comparing numerical values), and so forth.

The basic EMYCIN syntax for a rule is:

```
PREMISE:    ($AND(<clause1> . . . <clause-n>))
ACTION:     (CONCLUDE <new-fact> <CF>)
```

[3] These facts carry an associated degree of certainty also. The certainty is assumed to be 1.0 if the user answers a question without qualifying it. When a fact is concluded by a rule, its certainty is a function of (a) the certainty of the facts in the premise of the rule and (b) the strength of inference associated with the rule itself. This is described in more detail in the section on plausible inference. The important point here is not how the numbers are used, but that they are needed.

where the "$" prefix indicates that the premise is not a logical conjunction, but a plausible conjunction that must take account of the certainty factors associated with each of the clauses. The action taken by this rule is merely the addition of a new fact to the database (or, if the fact were already present, a modification of its certainty). EMYCIN also provide some mechanisms to allow the execution of more complicated actions. For example, in MYCIN we find the following rule (stated in English):

```
                              ----------
RULE160
-------

 If:   1) The time frame of the patient's headache is acute,
       2) The onset of the patient's headache is abrupt, and
       3) The headache severity (using a scale of 0 to 4; maximum is 4)
          is greater than 3
 Then:   1) There is suggestive evidence (.6) that the patient's
            meningitis is bacterial,
         2) There is weakly suggestive evidence (.4) that the patient's
            meningitis is viral, and
         3) There is suggestive evidence (.6) that the patient has blood
            within the subarachnoid space
                              ----------
```

Thus, this rule has three conclusions. It is represented internally in LISP as follows:[4]

```
                              ----------

PREMISE :   ($AND (SAME CNTXT HEADACHE-CHRONICITY ACUTE)
                  (SAME CNTXT HEADACHE-ONSET ABRUPT)
                  (GREATERP* (VAL1 CNTXT HEADACHE-SEVERITY 3)))

ACTION :    (DO-ALL (CONCLUDE CNTXT MENINGITIS BACTERIAL-MENINGITIS
                              TALLY 600)
                    (CONCLUDE CNTXT MENINGITIS VIRAL-MENINGITIS
                              TALLY 400)
                    (CONCLUDE CNTXT SUBARACHNOID-HEMORRHAGE YES
                              TALLY 600))
                              ----------
```

These examples illustrate the basic techniques for representing facts and knowledge within the EMYCIN framework. Similar examples could be given for each of the several framework systems that have been developed to facilitate the construction of rule-based expert systems, including:

[4] Since EMYCIN rules have a very regular syntax, a simple procedure can be used to translate rules from their internal format into English. It may be of interest to note that the list structures form the premise and the action are stored as values of the two properties, PREMISE and CONCLUSION, on the property list of the atom RULE160. The variable CNTXT is bound to the current object, in this case *the patient*. SAME and GREATERP* are matching predicates used to compare the values of the named attributes of the current object against the named value. The variable TALLY refers to the certainty values, which are given on a -1000 to 1000 scale for convenience.

OPS	Carnegie-Mellon University [Forgy77]
EMYCIN	Stanford University [vanMelle80]
AL/X	University of Edinburgh
EXPERT	Rutgers University [Weiss79a]
KAS	SRI International [Reboh81]
RAINBOW	IBM Scientific Center (Palo Alto) [Hollander79]

These framework systems provide important tools (such as editors) and facilities (such as explanation systems) that are beyond the scope of this paper to discuss. They also vary considerably in the syntax and the rule interpreters they employ. For example, in some of them all attributes must be binary. In some, uncertainty is expressed more formally as probabilities, or less formally as "major" or "minor" indicators, or cannot be expressed at all. And in some, additional structure is imposed on the rules to guide the rule interpreter. Despite these variations, these systems share a commitment to rules as the primary method of knowledge representation. This is at once their greatest strength and their greatest weakness, providing uniformity and modularity at the cost of imposing some very confining constraints.

2.2 Alternatives to Rule-Based Representation of Knowledge

There are alternatives to representing task-specific knowledge in rules. Naturally, it is sometimes advantageous to build a new system in PASCAL, FORTRAN, APL, BASIC, LISP, or other language, using a variety of data structures and inference procedures, as needed for the problem. Coding a new system from scratch, however, does not allow concentrating primarily on the *knowledge* required for high performance. Rather, one tends to spend more time on debugging the procedures that access and manipulate the knowledge.

The nature of the task sometimes requires more flexibility or more rigor than the rule-based frameworks provide. In those cases the frameworks mentioned below may provide satisfactory starting points. It should be noted that all of these frameworks easily allow specification of conditional rules; a sign of the times, however, is that the most restrictive frameworks (rule-based) arc currently the easiest to use.

There has been some experimentation with mixed representation as well [Aikins80, Reinstein81]. The basic idea is to increase the breadth of what one can represent easily while maintaining the advantages of having stylized representations (albeit more than one).

2.2.1 Frame-Based Representation Languages

One approach to representing knowledge that allows rich linkages between facts is a generalization of *semantic nets* [Brachman77] known as *frames* [Minsky75], A frame is an encoding of knowledge about an object, including not only properties (often called. *"slots")* and values, but pointers to other frames and attached procedures for computing values. The pointers indicate semantic links to other concepts, e.g., *brother-of*, and also indicate more general concepts from which properties may be inherited and more specialized concepts to which its properties will be manifested. Programming with this mode of representation is sometimes called *object-centeredprogramming* because knowledge is tied to objects and classes of objects.

Some well-known frame-based representation languages are

KRL	Xerox PARC [Bobrow77]
OWL	M.I.T. [Szolovits77]
UNITS	Stanford University [Stefik79]
FRL	M.I.T. [Roberts77]
AIMDS	Rutgers University [Sridharan80]
KL-ONE	Bolt, Beranek & Newman [Brachman78]

2.2.2 Logic-Based Representation Languages

A logic-based representation scheme is one in which knowledge about the world is represented as assertions in logic, usually first-order predicate logic or a variant of it. This mode of representation is normally coupled with an inference procedure based on theorem proving. Logic-based languages allow quantified statements and all other well-formed formulas as assertions, The rigor of logic is an advantage in specifying precisely what is known and knowing how the knowledge will be used. A disadvantage has been dealing with the imprccisidn and uncertainty of plausible reasoning.

To date there have been few examples of logic-based expert systems, in part because of the newness of the languages. Some logic-based representation languages are:

PLANNER	M.I.T.[Hewitt72]
PROLOG	Edinburgh University [Warren77]
ALICE	University of Paris [Lauriere78]
FOL	Stanford University [Weyhrauch80]

2.2.3 Generalized Languages

There is research in progress on general tools for helping a designer construct expert systems of various sorts. Designers specify the kind of representation and control and then add the task-specific knowledge within those constraints. The main advantage of such an approach is freedom -- designers specify their own constraints. The main disadvantage is complexity -- designers must be very knowledgeable about the range of choices and must be very patient and systematic about specifying choices. These tools look even more like high-level programming languages, which they are. The best known are:

ROSIE	Rand Corp [Fain81]
AGE	Stanford University [Nii79]
RLL	Stanford University [Greiner80]
HEARSAY-III	USC/ISI [Erman81]
MRS	Stanford University [Genesereth81a]

2.3 Knowledge Representation Issues

Regardless of the particular choice of representation language, a number of issues are important in the construction of knowledge bases for expert systems. We mentioned extendability, simplicity and explicitness as three global criteria. In addition the issues of consistency, completeness, robustness and transparency are major design considerations for all systems. For specific problems, it may be essential to represent and reason with temporal relations, spatial models, compound objects, possible worlds, beliefs, and expectations. These are discussed below.

Consistency in the knowledge base is obviously desirable. Because much of the knowledge coming from an expert is previously uncodified, and much of it comes with uncertainty, however, it is unrealistic to assume that the knowledge base can bc sufficiently cleansed to withstand a logician's scrutiny. In rule-based systems, there are syntactic checks made at the time new rules are entered to see if there is potential conflict between a new rule and existing rules. For example, two rules with the same premises but with different conclusions may be incompatible if the conclusions arc mutually exclusive. On the other hand, there arc many such pairs of rules that are perfectly acceptable because both conclusions are warranted. In MYCIN, for example, we find the same evidence "pointing to" different causes of an infection, with other rules invoking additional evidence to discriminate among likely causes.

Syntactic Completeness of the representation language is a logical requirement that many rule-based languages fail to satisfy. There are assertions, e.g., quantified statements, that are difficult or impossible to express. In DENDRAL, for example, it was difficult to express the proposition that if there exists a pair of data points bearing a complex relationship to one another then they constitute evidence for one class of interpretations of the data [Lindsay80].

Semantic Completerzess of the knowledge base for a problem area is also desirable. Because of the nature of the knowledge base and the way it is built, however, it will almost certainly fail to cover some interesting (sometimes important) possibilities. In a very narrow problem area, for example, there may be 100 attributes of interest, with an average of 4 important values for each attribute. (Only in extreme cases will all attributes be binary.) Thus there would be 79,800 possible rules relating two facts (400 items taken two at a time), over 10 million possible rules relating three facts, and so on. While most are semantically implausible, e.g., because of mutually exclusive values, the cost of checking all combinations for completeness is prohibitive. Checking the inferences made by a system in the context of carefully chosen test cases is currently the best way to check the completeness of coverage of the rules.

Precision in specialized domains is achievable for many of the facts and rules, but not all. There is a temptation to make overly-precise assertions for the knowledge base, even though there is no justification for fine precision. For example, although there are hospital-specific statistics about the incidence of a disease, one has to extrapolate to other (or all) hospitals very carefully. Representing degrees of imprecision is an important part of every representation methodology.

Default knowledge is important protection against incompleteness. If you know that devices manufactured by XYZ Corp are generally very reliable, then you might assume that the XYZ disk drive in your system is not the source of problems if you have no evidence to the contrary. Frame-based methods are designed to use their inheritance mechanisms to propagate default knowledge through parent-daughter links. In a rule-based system, default knowledge can certainly be represented but generally requires explicitly stating defaults for each class of actions.

Causal models provide a detailed specification of how a complex device works, whether it bc biological or mechanical. For man-made devices, the models can be made more precise. Representing something like a circuit diagram, and reasoning with it, is difficult although there have been successful research projects in which causal knowledge is central.

Temporal relations, as causal ones, are still difficult to represent and USC in generally satisfactory ways. Again, there has been good research, but expert systems generally do not reason well with continuous flows of data, or about continuous processes.

Strategies for problem solving are an important part of expertise. However, they are difficult to represent and use efficiently. Strategies are discussed in more detail in the section on control below.

The current state of the art of rcprcscnting knowledge about technical domains is adequate for many simple problems but requires considerably more research on these major issues, and more. (For comparisons among systems see [Brooks81, Ennis82].) Although rule-based frameworks have been used successfully for building several expert systems, the limitations are significant enough that researchers everywhere are looking for extensions or alternatives.

3 INFERENCE METHODS IN EXPERT SYSTEMS

3.1 Logical and Plausible Inference

Although the performance of most expert systems is determined more by the amount and organization of the knowledge possessed than by the inference strategies employed, every expert system needs inference methods to apply its knowledge. The resulting deductions can be strictly logical or merely plausible. Rules can be used to support either kind of deduction. Thus, a rule such as

```
Has(x, feathers) OR (Able(x, fly) & Able(x, lay-eggs)) -->
                    Class(x, bird)
```

amounts to a definition, and can be used, together with relevant facts, to deduce logically whether or not an object is a bird. On the other hand, a rule such as

```
State(engine, won't turn over) & State(headlights, dim) -->
                 State(battery, discharged)
```

is a "rule-of-thumb" whose conclusion, though plausible, is not always correct.

Clearly, uncertainty is introduced whenever such a judgmental rule is employed. In addition, the conclusions of a logical rule can also be uncertain if the facts it employs arc uncertain. Both kinds of uncertainty are frequently encountered in expert systems applications. However, in either case we arc using the rule to, draw conclusions from premises, and there are many common or analogous issues. In this section we temporarily ignore the complications introduced by uncertainty, and consider methods for using rules when everything is certain.

In terms of the production-systems model of rule-based expert systems, this section is concerned with the rule interpreter. As we mentioned in Section 2, the rule interpreter for a production system for unrestricted problem solving may have to employ complicated procedures to handle such things as pattern matching, variable binding, rule selection, and backtracking. To simplify our problem further, we shall restrict our attention to simple deduction systems, programs whose actions arc essentially limited to adding new facts to the global database. Our intent is to describe the general characteristics of the commonly used rule interpreters without becoming entangled 'in . the detailed mechanisms they employ; control strategies for more general problem solving systems are described in [Nilsson80].

3.2 Control

In this section we describe three commonly used control strategies: (1) data-driven, (2) goal-driven, and (3) mixed. Since control concerns are procedural, we shall describe these strategies semi-formally as if they were programs written in a dialect of PASCAL, a "Pidgin PASCAL." It is important to note at the outset, however, that these procedures are formal, employing no special knowledge about the problem domain; none of them possesses an intrinsic power to prevent combinatorial explosions. This has led to the notion of incorporating explicitly represented control knowledge in the rule interpreter, and idea that we discuss briefly at the end of this section.

3.2.1 Data-Driven Control

With data-driven control, rules are applied whenever their left-hand-side conditions are satisfied. To use this strategy, one must begin by entering information about the current problem as facts in the database. The following simplified procedure, which we shall call "Respond," can then be used to execute a basic data-driven strategy.

```
Procedure Respond;
Scan the database for the set S of applicable rules;
While S is non-empty and the problem is unsolved do
    begin
    Call Select-Rule(S) to select a rule R from S;
    Apply R and update the database;
    Scan the database for the set S of applicable rules
    end.
```

Here we assume that a rule is applicable whenever there are facts in the database that satisfy the conditions in its left-hand side. If there are no applicable rules, there is nothing to be done, except perhaps to return to the user and ask him or her to supply some additional information. (And, of course, if the problem is solved, there is nothing more to do.)

If there is only one applicable rule, the obvious thing to do is to apply it. Its application will enter new facts in the database. While that may either enable or disable previously inapplicable rules, by our assumption it will never disable a previously applicable rule.

If there is more than one applicable rule, we have the problem of deciding which one to apply. Procedure

Select-Rule has the responsibility for making this decision. Diffcrcnt data-driven stratègies differ greatly in the amount of problem-solving effort they devote to rule selection. A simple and inexpensive stratcgy is to select the first rule that is encountered in the scan for S -- "doing the first thing that comes to mind." Unfortunately, unless the rules are favorably ordered, this can result in many useless steps. Elaborations intended to overcome such shortcomings can make data-driven control arbitrarily complex.

Data-driven control is very popular, as is evidenced by the fact that it is known by so many different names (bottom-up, forward chaining, pattern-directed, or antecedent reasoning). R1 is an excellent example of an expert system that employs this strategy [McDermott80]. The popularity of data-driven control derives largely from the fact that such a program can respond quickly to input from the user, rather than forcing the user to wait until the program gets around to what the user wants to talk about.

We have already mentioned the potential inefficiency of this strategy. Other problems that are often overlooked can arise for programs intended to be used by naive users. For example, as a data-directed program fires off one rule after another, its behavior can appear to be aimless, undermining a user's confidence in its soundness. Also, since the user must begin by entering a set of facts, some kind of a language is needed to convert facts as expressed in the user's terms into the appropriate internal representation; menu systems may provide an acceptable solution, but a need for greater flexibility is frequently encountered. Both of these problems can be circumvented by using goal-driven control.

3.2.2 Goal-Driven Control

A goal-driven control strategy focuses its efforts by only considering rules that are applicable to some particular goal. Since we are limiting ourselves to rules that can add simple facts to the database, achieving a goal G is synonymous with showing that the fact statement corresponding to G is true. In nontrivial problems, achieving a goal requires setting up and achieving subgoals. This can also lead to fruitless wandering if most of the subgoals are unachievable, but at least there is always a path from any subgoal to the original goal.

Suppose that the user specifies a goal statement G whose truth is to be determined -- typically a fact that the user would like to have present in the database. Then the following simplified procedure, which we shall call "Achieve," carries out a basic goal-driven strategy.

```
Procedure Achieve(G);
Scan the knowledge base for the set S of rules that determine G;
If S is empty then ask the user about G else
    While G is unknown and rules remain in S do
```

```
begin
Call Choose-Rule(S) to choose a rule R from S;
G' <-- condition(R):
If G' is unknown then call Achieve(G');
If G' is true then apply R
end.
```

Thus, the first step is to gather together all of the rules whose right-hand-sides can establish G. If there is more than one relevant rule, procedure Choose-Rule receives the problem of making the choice. Once a rule R is selected, its left-hand-side G' is examined to see if R is applicable. If there is no information in the database about G', the determination of its truth or falsity becomes a new subgoal, and the same procedure Achieve is applied to G' recursively.

The search continues in this fashion, working systematically backward from the original goal, until a subgoal is encountered for which there are no rules. At this point the system turns to the user and asks for the relevant facts. If the user cannot supply the needed information, then the rule the system was working on at the time cannot be used, but other lines of reasoning can still be explored. If the information supplied shows that G' is true, then R is applied. The process continues in this manner until either G is established to be true or false, or no applicable rules remain.

Since the left-hand-side of the selected rule becomes the next subgoal, the choice of a rule is equivalent to subgoal selection. Different goal-driven strategies differ greatly in the amount of effort they devote this problem. A simple and inexpensive strategy is to select the first rule that is encountered in the scan for S.[5] Unfortunately, unless the rules are favorably ordered, this can lead to exploring unpromising subgoals. As in the case of data-driven control, elaborations intended to overcome such shortcomings can make goal-driven control arbitrarily complex.

Goal-driven control has also been used in many systems, and is variously known as top-down, backward-chaining, or consequent reasoning. A primary virtue of this strategy is that it does not seek information and does not apply rules that are unrelated to its overall goal. Furthermore, as we have seen in the excerpt from a MYCIN session, a rule-based system using this strategy can provide illuminating explanations of its behavior merely by telling the user what goal it is working on and what rule it is using.

---

[5] MYCIN employs a slightly different strategy: instead of making a selection, MYCIN employs a cautious strategy of applying all of the rules in S. This is because individual MYCIN rules usually do not establish the truth of falsity of G, but merely increase or decrease its certainty; a final certainty assessment cannot be made until all of the rules are used.

Probably the chief disadvantage of a goal-driven strategy is that it does not allow 'the user to steer it by volunteering relevant information about the problem. This can make goal-driven control unacceptable when rapid, real-time response is required.

3.2.3 Mixed Strategies

Data-driven and goal-driven strategies represent two extreme approaches to control. Various mixtures of these. approaches have been investigated in an attempt to secure their various advantages while minimizing their disadvantages. The following simple procedure combines the two by alternating between the two modes.

```
Procedure Alternate;
Repeat
    Let user enter facts into global database:
    Call Respond to deduce consequences;
    Call Select-Goal to select a goal G;
    Call' Achieve(G) to try to establish G
 until the problem is solved..
```

Here Respond and Achieve are the data-driven and goal-driven procedures described previously. Select-Goal, which we do not attempt to specify, uses the partial conclusions obtained from the data-driven phase to determine a goal for the goal-driven phase. Thus, the basic idea is to alternate between these two phases, using information volunteered by the user to determine a goal, and then querying the user for more information while working on that goal.

A variant of this procedure is used in the PROSPECTOR program [Duda79]. In this case, Select-Goal uses a heuristic scoring procedure to rank the goals in a prespecified set of "top-level" goals, but the user is allowed to see the results and to make the final selection. Furthermore, a modified version of Achieve is used which ceases working on a goal whenever (a) its score drops and (b) it is not the highest-scoring top-level goal. Thus, PROSPECTOR works in a goal-driven mode when it seems to be making progress, but returns to the user for help in goal selection when serious trouble is encountered.

3.3 Explicit Representation of Control Knowledge

The advantages of making the task-specific knowledge modular and explicit extend to control knowledge as well. The strategy by which an expert system reasons about a task depends on the nature of the task and the nature of the knowledge the system can use. Neither data-driven, goal-driven, nor any particular mixed strategy is good for every problem. Different approaches are needed for different problems. Indeed, one kind of knowledge possessed by experts is knowledge of procedures that are effective for their problems.

In most expert systems, the control strategy is encoded in procedures much like the ones we exhibited in pseudo-PASCAL. Thus, control knowledge is not explicitly represented, but is buried in the code. This means that the system cannot easily explain its problem solving strategy, nor can the system builder easily modify it. Several interesting attempts have been made to extract this knowledge and represent it explicitly. In his work on TEIRESIAS, Davis included the use of meta-rules, rules that determined the control strategy [Davis77a]. TEIRESIAS essentially implements the procedure Select-Rule as a rule-based system. That is, strategic knowledge is used to reason about the most appropriate rules to invoke during problem solving or the most appropriate order in which to invoke them. Because the strategy rules can be context-specific, the result is a system that adapts its rule selection strategy to the nature of the problem. Other important work on explicit control of reasoning in expert systems can be found in [Aikins80, Barnett82, Clancey81, deKleer77, Genesereth81a, Georgeff82].

4 REASONING WITH UNCERTAINTY

The direct application of these methods of deduction to real-world problems is complicated by the fact that both the data and the expertise are often uncertain. This fact has led the designers of expert systems to abandon the pursuit of logical completeness in favor of developing effective heuristic ways to exploit the fallible and but valuable judgmental knowledge that human experts bring to particular classes of problems. Thus, we now turn to comparing methods that have been used to accommodate uncertainty in the reasoning.

4.1 Plausible Inference

Let A be an assertion about the world, such as an attribute-object-value triple. How can one treat the uncertainty that might be associated with this assertion? The classical formalism for quantifying uncertainty is probability theory, but other alternatives have been proposed and used. Among these are certainty theory, possibility theory, and the Dempster/Shafer theory of evidence. We shall consider all four of these approaches in turn, with emphasis on the first two.

4.2 Bayesian Probability Theory

With probability theory, one assigns a probability value P(A) to every assertion A. In expert systems applications, it is usually assumed that P measures the degree to which P is believed to be true, where P = 1 if

A is known to be true, and P = 0 if A is known to be false.[6] In general, the degree of belief in A will change as new information is obtained. Let P(A) denote our initial or prior belief in A, and let the conditional probability P(A|B) denote our revised belief in A upon learning that B is true. If this change in probability is due to the application of the rule B --> A in a rule-based system, then some procedure must be invoked to change the probability of A from P(A) to P(A|B) whenever this rule is applied.

In a typical diagnosis situation, we think of A as a "cause" and B as an "effect," and view the computation of P(A|B) as an inference that the cause is present upon observation of the effect. The expert often finds it easier to estimate P(B|A) -- the prdbability of observing the effect B when the cause A is active. In medical situations, this is further justified by the argument that the probability of a disease given a symptom may vary with time and place, while the probability of a symptom given a disease remains invariant [Lusted68]. Thus, *Bayes' Rule* is commonly employed to compute P(A|B) from P(B|A).

It turns out that the important information that is needed to employ Bayes' Rule is the prior probability P(A) and the likelihood ratio L defined by

$$L = \frac{P(B \mid A)}{P(B \mid \sim A)} ,$$

where P(B|~A) is the probability of observing effect B when cause A is absent.

If we think of the link between B and A as being expressed by a rule of the form B --> A, then we can think of *the logarithm* of the likelihood ratio L as representing the *strength* or *weight* of the rule; rules with positive weights increase the probability of A, and rules with negative weights decrease it.

Two generalizations are needed to employ this result in practice. First, we need a way to combine the effects of several rules, particularly when they are in conflict. Second, we need a way to employ uncertain evidence. While a thorough treatment of these topics is beyond the scope of this paper, it is useful to explore this topic sufficiently to reveal the problems that are encountered and the general nature of their solutions.

---

[6] Traditionally, the probability P(A) is interpreted to measure the frequency of occurrence of the event A in a series of random trials. With this frequency-ratio interpretation, P is called an objective probability, and the estimation of a numerical value for P is a statistical problem. When P(A) is used to measure degree of belief, it is called a *subjective* probability, and the estimation of numerical values is done by interviewing experts [Savage71]. However, in either case the same calculus is used for combining probabilities, and a frequency-ratio interpretation is often used in practice to estimate subjective probabilities and vice versa. The distinctions between objective and subjective probability theory are treated in depth in [Fine73].

4.2.1 Combining Rules

Suppose that we have *n* plausible rules of the form

$$B_1 \dashrightarrow A,\ B_2 \dashrightarrow A,\ \ldots .\ B_n \dashrightarrow A,$$

each with its own weight. Formally, the generalization of Bayes' Rule is simple. We merely consider B to be the conjunction $B = B_1 \,\&\, B_2 \,\&\, \ldots \,\&\, B_n$, and use the likelihood ratio

$$L = \frac{P(B_1 \,\&\, B_2 \,\&\, \ldots \,\&\, B_n \mid A)}{P(B_1 \,\&\, B_2 \,\&\, \ldots \,\&\, B_n \mid \sim A)}$$

The problem with this solution is that it implies that we not only have weights for the individual rules connecting the B_i to A, but that we also have weights for the pairs B_i & B_j, triples B_i & B_j and B_k, and so on, not to mention combinations involving negations when the evidence is known to be absent. This not only leads to extensive, nonintuitive computations, not directly related to the rules, but also requires forcing the expert to estimate a very large number of weight values.

A major simplification can be achieved if it is possible to assume that the B_i are *statistically independent,* both when A is present (true) and when A is not present (false). In that case, the conditional probabilities shown above factor, and L simplifies to the product of the separate likelihood ratios. In other words, under that assumption, we need only have one weight associated with each rule, and we can combine the effects of several pieces of evidence by adding the separate weights.

In general, of course, this assumption is not justified. In fact, it can be shown that the assumption cannot be made repeatedly when there are multiple assertions.[7] In its extreme form, this approach suggests designing an expert system by gathering all the evidence that bears in any way on a final conclusion A and going in, one step from the observations to the conclusion. Such an approach typically founders on the complexity of the interactions among observations that are not even approximately independent.

The pragmatic solution to this problem places the responsibility on the knowledge engineer to see that the rules are properly structured. Many problems caused by interactions can be solved by employing a hierarchical structure, with several levels of assertions between the direct observations and the final conclusions. The goal is to localize and limit the interactions, and to have a relatively small number of clauses in a condition and a relatively small number of rules sharing a common conclusion. Note that this limitation on the number of rules does not reduce the amount of evidence considered in reaching a conclusion, but rather controls the ways in

---

[7] To be more specific, if $A_1, A_2, \ldots . A_m$ are m mutually exclusive and exhaustive assertions, then the 2m assumptions that the B_i are independent under A_j and $\sim A_j$ are inconsistent with the laws of probability theory if $m > 2$ (see [Pednault81]). Computations based on such assumptions will fail to show that A_1 must be true if A_2 through A_m have been ruled out.

which the observations are allowed to interact. A hierarchical structure is typically employed by the experts themselves to reduce the complexity of a problem. Wherever the remaining interactions still prevent the assumption of local independence, the rules have to be reformulated to achieve the desired behavior. For example, in the strongly interacting situation where B_1 suggests A and B_2 suggests A, but the simultaneous presence of both B_1 and B_2 rules out A one may have to augment the rule set

```
{ (B1 --> A with weight L1)
  (B2 --> A with weight L2) }
```

with the rule (B_1 & B_2 --> A with weight-m). Thus, rather than viewing probability theory as a paradigm that prescribes how information should be processed, the knowledge engineer employs it as a tool to obtain the desired behavior.

4.2.2 Uncertain Evidence

There are two reasons why an assertion B might be uncertain: (1) the user might have said that B is uncertain, or (2) the program might have deduced B using a plausible rule. If we want to use B in a rule to compute P(A|B), the question then arises as to how to discount the conclusion because of the uncertainty associated with B.

Let E symbolize whatever evidence was used to determine B, and let P(B|E) denote the current probability of B. Then our problem is to compute P(A|E), the current probability of A given this same evidence. It is shown in [Duda76] that under certain reasonable assumptions we should be able to compute P(A|E) by the formula

$$P(A|E) = P(A|B)*P(B|E) + P(A|\sim B)*[1 - P(B|E)] .$$

This formula certainly works in the extreme cases of complete certainty. That is, if we know that B is true we obtain P(A|B), and if we know that B is false we obtain P(A|~B). Unfortunately, a serious problem arises in intermediate cases. In particular, suppose that E actually supplies no information about B, so that P(B|E) is the same as the prior probability P(B). While the formula above promises to yield the prior probability P(A), when the computation is based on numerical values obtained from the expert, the resulting value for P(A|E) will usually not agree with the expert's estimate for the prior probability P(A). That is, the four quantities P(A), P(B), P(A|B) and P(A|~B) arc not independent, and the expert's subjective estimates for them arc almost surely numerically inconsistent.

In this particular case, the problem can be solved in various ways (such as by not asking the expert for P(A), but by computing it from P(A) = P(A|B)P(B) + P(A|~B)P(~B)). However, that only makes the parameters for one rule consistent, and the solution is not at all obvious when there is a network of rules that have inconsistent values for probability parameters. The solution adopted in the PROSPECTOR system was to replace the above expression for $P(A|E)$ by a piecewise linear function of $P(B|E)$ that yields the expert's estimate for P(A) when $P(B|E)$ is numerically equal to the expert's estimate for P(B) (see [Duda76]). Interestingly, the resulting formulas turn out to be closely related to the certainty measure used in MYCIN, which we consider next.

4.3 Certainty Theory

We have seen several problems that arise in using traditional probability theory to quantify uncertainty in expert systems. Not the least of these is the need to specify numerical values for prior probabilities. While an expert may be willing to say how certain he or she feels about a conclusion A when evidence B is present, he or she may be most reluctant to specify a probability for A in the absence of any evidence, particularly when rare but important events are involved. Indeed, some of the problems that are encountered in obtaining consistent estimates of subjective probabilities may well be due to the fact that the expert is not able to separate probability from utility or significance, and is really expressing some unspecified measure of importance.

To accommodate this reality, the designers of the MYCIN system developed a theory of certainty that provides a simple and effective alternative approach [Shortliffe75]. The central notion is that we can associate a certainty measure C(A) with every assertion A, where C = 1 if A is known to be true, C = -1 if A is known to be false, and C = 0 if nothing is known about A.[8] A similar certainty factor CF is associated with every rule. The theory consists of procedures for updating certainties as rules are applied, and an analysis of the properties of these procedures.

The procedures for updating certainties are easily stated. Initially, the certainty associated with any assertion is 0. If a rule says that B --> A with a certainty factor CF, then the certainty of A is changed to CF when B is observed to be true. Only two things remain to be specified: (1) the procedure for combining evidence when more than one rule concludes A, and (2) the treatment of uncertain evidence. We consider each of these in turn.

4.3.1 Corn bining Evidence

Suppose that (1) the present certainty of an event A is CA (which may be non zero because of the previous application of rules that conclude A), (2) there is an unused rule of the form B --> A with a certainty factor CF, and (3) B is observed to be true. Then the EMYCIN formula for updating C(A) to $C(A|B)$ is

---

[8]If we also assume that we can associate probabilities with assertions, then it would appear that C = 1 corresponds to P = **1, C = -1** corresponds to P = 0, and C = 0 corresponds to P being at its prior value. The original MYCIN definitions of certainty were in terms of piecewise-linear functions of probability. However, the EMYCIN view, which we present in this section, is that the calculus of certainty factors is a heuristic approach that allows rule-based systems to cope with uncertainty in a simple and straightforward way, judging it more by its usefulness than by its theoretical properties [vanMelle80]. It is a one-number calculus that combines subjective estimates of probability and risk in a measure of *importance*, which bears no simple relationship to probability alone.

$$C(A|B) = \begin{cases} CA + CF - CA * CF & \text{if } CA \text{ and } CF > 0 \\ CA + CF + CA * CF & \text{if } CA \text{ and } CF < 0 \\ \dfrac{CA + CF}{1 - \min\{|CA|, |CF|\}} & \text{otherwise}. \end{cases}$$

This is the EMYCIN analog of the procedure of multiplying likelihood ratios to combine "independent" evidence. By applying it repeatedly, one can combine the conclusions of any number of rules B_1 --> B_2 --> A, ... B_n --> A. Aside from being easy to compute, it has several other desirable properties. First, the resulting certainty C(A|B) always lies between -1 and 1, being +1 if CA or CF is +1, and -1 if CA or CF is -1. When contradictory conclusions are combined (so that CA = -CF), the resulting certainty is 0. Except at the singular points (1, -1) and (-1, 1), C(A|B) is a continuous function of CA and CF, increasing monotonically in each variable. The formula is symmetric in CA and CF, and the results it yields when more than two pieces of evidence are combined are independent of the order in which they are considered.

Of course, none of these properties prove that this is the "correct" way to combine the conclusions of rules. Indeed, the results will be wrong in strongly interacting cases, such as our previous example in which B_1 suggests A and B_2 suggests A, but the simultaneous presence of B_1 and B_2 rules out A. As with the use of Bayesian methods, the knowledge engineer should view certainty theory as a tool to be employed to produce the desired behavior.

4.3.2 Uncertain Evidence

When the evidence B for a rule B --> A is itself uncertain, it is clear that the strength of the conclusion must be reduced. The EMYCIN procedure is to multiply the certainty factor CF for the rule by the certainty of B, provided that the certainty of B is positive. If the certainty of B is negative, the rule is considered to be inapplicable, and is not used.[9] EMYCIN assumes that a rule cannot be employed unless the certainty of its antecedent is greater than a threshold value of 0.2. This heuristic -- which implies that the certainty of a conclusion is not a strictly continuous function of the certainty of the evidence -- saves time by inhibiting the application of many marginally effective rules, and saves confusion by making explanations provided by the system more understandable.

One more effect of uncertain evidence remains to be mentioned. In general, the antecedent B of a rule is a logical function of predicates or relations on associative triples. Since these functions or relations can return certainty values rather than truth values, there is a question as to how the certainty of their logical combination is determined. The answer is that it is computed through the recursive application of the following formulas:

[9] If the absence of B has a significant effect on A, we could add a rule of the form -B --> ~A with some certainty factor CF'.

$$C(B_1 \text{ OR } B_2) = \max\{C(B_1), C(B_2)\}$$

$$C(B_1 \,\&\, B_2) = \min\{C(B_1), C(B_2)\}$$

$$C(\sim B) = -C(B).$$

These formulas are essentially the same as the corresponding formulas of possibility theory, which is discussed briefly in the next section.

4.4 Possibility Theory

Probability theory captures only some of the important aspects of uncertainty, and a variety of alternative approaches, such as certainty theory, have been developed to overcome its limitations. One of the most interesting of the recent alternatives is Zadeh's theory of possibility [Zadeh78]. It is based on his earlier development of the theory of fuzzy sets [Zadeh65], much as probability theory is based on measure theory.

Zadeh asserts that although the random model of probability theory may be appropriate for problems involving the measure of information, it is inappropriate for problems concerning the meaning of information. In particular, much of the uncertainty surrounding the use of English terms and expressions concerns vagueness rather than randomness. Possibility theory provides a formalism for treating vagueness that is analogous to probability theory as a formalism for treating randomness.

The theory of fuzzy sets expresses this kind of imprecision quantitatively by introducing characteristic or membership functions that can assume values between 0 and 1. Thus, if S is a set and ifs is an element of S, a fuzzy subset F of S is defined by a membership function $\mu F(s)$ that measures the degree to which s belongs to F. To use a standard example, if S is the set of positive integers and F is the fuzzy subset of small integers, then we might have $\mu F(1) = 1$, $\mu F(2) = 1$, $\mu F(3) = .8, \ldots,$ $\mu F(20) = .01$, and so on. Let X be a variable that can take values in S. The statement "X is F" (for example, the statement "X is a small integer"), induces a possibility distribution on X, and the possibility that X = s is taken to be $\mu F(s)$.

Now probability theory is not concerned with how the numerical values of probabilities are determined, but rather with the rules for computing the probability of expressions involving random variables. Similarly, possibility theory is not concerned with how the numerical values of the possibility distributions are obtained, but rather with the rules for computing the possibilities of expressions involving fuzzy variables. In particular, if Poss{X = s} is the possibility that the fuzzy variable X is equal to s, then the formulas for disjunction, conjunction and negation are

$$\text{Poss}\{X=s \text{ OR } Y=t\} = \max[\text{Poss}\{X=s\}, \text{Poss}\{Y=t\}]$$

$$\text{Poss}\{ x=s \ \& \ Y=t \} \quad = \quad \min[\ \text{Poss}\{X=s\}, \text{Poss}\{Y=t\}\ ]$$

and

$$\text{Poss}\{ x \neq s \} \quad = \quad 1 - \text{Poss}\{X=s\}.$$

For most of the concepts of probability theory there is a corresponding concept in possibility theory. For example, it is possible to define multivariate possibility distributions, marginal possibility distributions, and conditional possibility distributions (see [Zadeh78]). Thus, in principle one can use fuzzy possibility theory much like probability theoiy to quantify the uncertainty introduced by vagueness, whether the vagueness comes from the data or from the rules.

Although possibility theory is a subject of great interest, it has yet to be exploited in work on expert systems. This is partly due to the fact that most of the problems that limit probability theory also arise in possibility theory -- such as the problem of prior possibilities and the problem of dependence in multivariate possibility distributions. Furthermore, as with certainty theory, possibility theory suffers from the problem that the semantics of its measure are not objectively defined. However, the distinction between uncertainty due to randomness and uncertainty due to vagueness is both valid and important, and should play a role in work in expert systems.

4.5 The Dempster/Shafer Theory of Evidence

We conclude this overview of formalisms for treating uncertainty with a brief consideration of a generalization of probability theory created by Dempster and developed by Shafer that has come to be known as the Dempster/Shafer theory of evidence [Shafer76, Barnett81].

Dempster and Shafer insist that one must make a fimdamental distinction between uncertainty and ignorance. In probability theory, one is forced to express the extent of one's knowledge about or belief in an assertion A through a single number, P(A). Dempster and Shafer point out that the classical Bayesian agony concerning prior probabilities is often due to the fact that one often simply does not know the values of prior probabilities, and this ignorance may make any particular choice arbitrary and unjustifiable.

The Dempster/Shafer theory of evidence recognizes the distinction between uncertainty and ignorance by introducing belief functions that satisfy axioms that are weaker than those of probability functions. Thus, probability functions are a subclass of belief functions, and the theory of evidence reduces to probability theory when the probability values are known. Roughly speaking, the belief functions allow us to use our knowledge to put constraints or bounds on the assignment of probabilities to events without having to specify the probabilities themselves. In addition, the theory of evidence provides appropriate methods for computing belief functions for combinations of evidence.

As one might expect, a theory that includes probability theory as a special case suffers from many of the same problems that plague probability theory. The greater complexity results in an increase in computational problems as well, and the conclusions that can be reached are necessarily weaker. However, when available. knowledge does not justify stronger conclusions, this latter fact has to be accepted. Whether or not the theory of evidence will provide the basis for computationally effective procedures for treating uncertainty, it deserves attention for exposing the effects of lack of knowledge on probabilistic reasoning.

5 KEY CONCEPTS

In the previdus three sections we focused on three central issues in the design of expert systems, with special attention to rule-based systems. The representation, inference methods and methods for reasoning under uncertainty are the elements of the design of rule-based systems that give them power. We turn now to a broader look at several less technical aspects of building an expert system. These are observations derived from our own experience and constitute suggestions for designing an expert system. They also reflect the current state of the art.

In this section we first list several of these considerations, with very little explanation. Then we look at the nature of the problem (the first of the considerations) in more detail from three different perspectives: the types of problems for which expert systems have been developed (Sec.5.1), the nature of the data encountered in these problems (Sec.5.2), and the nature of the expertise (Sec. 5.3).

We spoke earlier of the importance of separating task-specific knowledge from a system's inference methods, and we discussed the representation and inference methods by which we can realize the truth in the assumption that "knowledge is power." We list here some of those and other key ideas in putting together an expert system.

NATURE OF THE PROBLEM:

Narrow scope -- The task for the system must be carefully chosen to be narrow enough that the relevant expertise can be encoded, and yet complex enough that expertise is required. This limitation is more because of the time it takes to engineer the knowledge into a system including refinement and debugging, than because space required for the knowledge base.[10]

[10] This contrasts with early work in AI in which space was at least as much an issue.

Existence of an expert -- Thcie are problems so new or so complex that no one ranks as an expert in the problem area. Generally speaking, it is unwise to expect to be able to construct an expert system in areas where there are no experts.

Agreement among experts -- If current problem solving expertise in a task area leaves room for frequent and substantial disagreements among experts, then the task is not appropriate for an expert system.

Data available -- Not only must the expertise be available, but test data must be available (preferably online). Since an expert system is built incrementally, with knowledge added in response to observed difficulties, it is necessary to have several test cases to help explore the boundaries of what the system knows.

Milestones definable -- A task that can be broken into subtasks, with measurable milestones, is better than one that cannot be demonstrated until *all the* parts arc working.

REPRESENTATION:

Separation of task-specific knowledge from the rest of the program -- This separation is essential to maintain the flexibility and understandability required in expert systems.

Attention to detail -- Inclusion of very specific items of knowledge about the domain, as well as general facts, is the only way to capture the expertise that experience adds to textbook knowledge.

Uniform data structures -- A homogeneous representation of knowledge makes it much easier for the system builder to develop acquisition and explanation packages.

INFERENCE:

Symbolic reasoning - It is commonplace in AI, but not elsewhere, to regard symbolic, non-numeric reasoning as a powerful method for problem solving by computers. In applications areas where mathematical methods are absent or computationally intractable, symbolic reasoning offers an attractive alternative.

Combination of deductive logic and plausible reasoning -- Although deductive reasoning is the standard by which we measure correctness, not all reasoning -- even in science and mathematics -- is accomplished by deductive logic. Much of the world's expertise is in heuristics, and programs that attempt to capture expert-level knowledge need to combine methods for deductive and plausible reasoning.

Explicit problem solving strategy -- Just as it is useful to separate the domain-specific knowledge from the inference method, it is also useful to separate the problem solving strategy from both. In debugging the system it helps to remember that the same knowledge base and inference method can produce radically different behaviors with different strategies. For example, consider the difference between "find the best" and "find the first over threshold".

Interactive user interfaces -- Drawing the user into the problem solving process is important for tasks in which the user is responsible for the actions recommended by the expert system, as in medicine. For such tasks, the inference method must support an interactive style in which the user contributes specific facts of the case and the program combines them in a coherent analysis.

EXYLANATION:

Static queries of the knowledge base -- The process of *constructing* a large knowledge base requires understanding what is (and is not) in it at any moment. Similarly, *using* a system effectively depends on assessing what it does and does not know.

Dynatnic queries about the line of reasoning -- As an expert system gathers data and makes intermediate conclusions, users (as well as system builders) need to be able to ask enough questions to follow the line of reasoning. Otherwise the system's advice appears as an oracle from a black box and is less likely to be acceptable.

KNOWLEDGE ACQUISITION:

Bandwidth -- An expert's ability to communicate his/her expertise within the framework of an expert system is limited by the restrictions of the framework, the degree to which the knowledge is already well-codified, and the speed with which the expert can create and modify data structures in the knowledge base.

Knowledge engineer -- One way of providing help to experts during construction of the knowledge base is to let the expert communicate with someone who understands the syntax of the framework, the rule interpreter, the process of knowledge base construction, and the practical psychology of interacting with world-class experts. This person is called a *"knowledge engineer"*.

VALIDATION:

Level of perfornance -- Empirical measures of adequacy are still the best indicators of performance, even though they arc not sufficient for complete validation by any means. As with testing new drugs by the pharmaceutical industry, testing expert systems may best be accomplished by randomized studies and double-blind experiments.

Static evaluation -- Because the knowledge base may contain judgmental rules as well as axiomatic truths, logical analysis of its completeness and consistency will be inadequate. However, static checks can reveal potential problems, such as one rule subsuming another and one rule possibly contradicting another. Areas of weakness in a knowledge base can sometimes be found by analysis as well.

5.1 Classes of Problems for Expert Systems

The first of the key concepts listed above was the nature of the problem. We examine this issue in somewhat more detail in this and the next two sections. While there are many activities an expert performs, the activities for which expert systems have been built fall into three categories: analysis, synthesis, and interface problems.

Analysis Problems have been the most successfully solved with the knowledge engineering approach to date. Many applications programs that have the characteristics of expert systems have been developed for analysis problems in a diversity of areas including: chemistry [Buchanan78, Carhart79]; genetics [Stefik78]; protein crystallography [Engelmore79]; physics [Bundy79, Larkin80, Novak80,]; interpretation of oil well logs [Barstow79b, Davis81]; electronics troubleshooting [Addis80, Bennett81b, Brown82, Davis82b, Genesereth81b, Kandt81, Stallman77]; materials engineering [Basden82, Ishizuka81]; mathematics [Brown78, Moses71]; medical diagnosis [Chandrasekaran80, Fagan80, Gorry78, Heisdr78, Horn81, Kaihara78, Lindberg81, Pati181, Pople77, Reggia78, Shortliffe76, Shortliffe81, Swartout77, Szolovits78, Tsotsos81, Weiss79b]; mineral exploration [Duda79]; aircraft identification and mission planning [Engelman79]; military situation assessment [McColl79, Nii82]; and process control [Mamdani82]. Within these and other disciplines, analysis problems are described using many different terms, including:

- Data Interpretation
- Explanation of Empirical Data
- Understanding a Complex of Data (c.g., signal understanding)
- Classification
- Situation Assessment
- Diagnosis (of diseases, equipment failures, etc.)
- Troubleshooting
- Fault Isolation
- Debugging
- Crisis Management (diagnosis half)

An expert system working on one of these problems analyzes a description of a situation, and provides plausible interpretations of what the data seem to indicate. The data may come from a variety of sources ranging from subjective opinion to precise readings of instruments.

Synthesis Problems have the character of constructing a solution to satisfy a goal within stated constraints. In many cases, solutions to small, local problems need to be synthesized into a coherent solution that satisfies global constraints. Synthesis problems arise in many fields including: planning cxpcrimcnts in molecular gcnctics [Friedland79, Stefik80], configuring the components of a computer system [McDcrmott80, McDcrmott81]; scheduling [Fox82, Goldstein79, Lauriere78]; automatic programming [Barstow79a, McCune77]; electronics design [dcKlccr80, Dincbas80, Sussman78], and chemical synthesis [Gelernter77, Wipke77]. These problems have been called:

- Planning (or Constructing a Plan of Action)
- Fault Repair
- Process Specification
- Design (of complex devices or of experiments)
- Configuration
- Therapy (or therapy planning)
- Automatic Programming
- Computer-Aided Chemical Synthesis Planning

In addition to analysis and synthesis problems, expert systems have been built to provide advice on how to usc a complex system [Anderson76, Bennett79, Gencscreth78, Hewitt75, Krueger81, Rivlin80, Waterman791 or to tutor a novice in the use or understanding of a body of knowledge [Brown82, Clancey79, O'Shea79]. These problems arc partly analytic, since the advice or tutorial must be guided by an analysis of the context, and partly synthetic since the advice must bc tailored to the user and the problem at hand.

5.2 The Data

One of the central concerns in choosing a task for an expert system is the nature of the data. In problems of analysis, the data require interpretation by means of some model or theory. Yet in many interesting problems, the data are not as complete or "clean" as the theory seems to require. In applying a theory to individual cases, the data are not always available to "plug into" formulas and principles. In the absence of perfect data, however, experts can still provide good suggestions, when a novice can not. We have identified several important concerns, briefly discussed below: incompleteness, noise and non-independence.

Incompleteness of the data is a common difficulty. In medical diagnosis, for example, a physician usually must act before all possibly relevant tests have been made. Uncertainty of the data compounds the difficulty. Decision makers know that their sources of information arc fallible, whether the sources are instruments or other persons. Some tests arc notoriously unreliable, some items of information arc so incongruous with other data that something must be wrong. Yet, in the face of these uncertainties in the data, expert decision makers can still integrate the results of many tests better than novices.

Noise in the data can bc confusing. Spurious data points, or "red herrings", can throw the best problem solvers off the track. However, experts have had more cxpcricncc in sorting out good and bad data and arc less likely to remain confused than novices. The data given to a decision maker can bc noisy for a variety of reasons including electronic noise, misreading dials and indicators, and transcription errors. By the time the decision maker sees the data, it is often too late to check the validity of any single data point.

Non-independence in the data is often a difficulty, particularly for statistical methods that rely on assumptions of independence to combine pieces of evidence. In most interesting problems, though, there are processes linking many parts of complex systems, so that evidence about one part of the system is richly linked with other pieces of evidence. If the data were known to be error-free, then avoiding redundancy would simplify the decision making process. However, in the face of possibly unreliable data, redundancy is beneficial in helping reduce the effects of spurious data.

The data are often of uneven grain size, combining gross descriptive reports with minute, precise statements of fact. Qualitative and quantitative information is mixed. Subjective reports are mixed with objective statements. There is no uniform theoretical framework in which information at all these levels can be combined. Yet, decision makers faced with less than perfect data often welcome more information, regardless of how heterogeneous it is. The volume of data, however, can get to be confusing. The combinatorics of considering meaningful clusters of data quickly swamps a person's ability to consider combinations of data points systematically.

One of the primary advantages of an expert system in coping with all of this ambiguity is its ability to exploit redundancy. Multiple pieces of data can indicate more or less the same intcrprctation, some more strongly than others, while others indicate mutually exclusive interpretations. An expert system will work with the data available using the overlapping contributions to help make up for missing data and incomplete interpretation rules.

5.3 The Expertise

The proficiency of an expert system is dependent on the amount of domain-specific expertise it contains. But expertise about interesting problems is not always neatly codified and waiting for transliteration into a program's internal representation. Expcrtisc exists in many forms and in many places, and the task' of *knowledge engineering* includes bringing together what is known about a problem as well as transforming (not merely transcribing) it into the system.

We have already said that much of the expertise is symbolic, heuristic!, and not well formalized. That implies that an expert's knowledge is not always certain, that it is provisional, without guarantees of correctness. Bccausc it is not well formalized (e.g., in neat theoretical formulations in textbooks), a specialists's knowledge is not always easily accessible. In addition, we have to assume that it is incomplete, since the facts and heuristics change with increased experience.

Because of the multitude of sources of expertise, an expert articulates what he or she knows in a mixture of frameworks, using terminology ranging from broad notions of common sense to precisely defined theoretical terms. As with the data, there is a mixture in concepts that the knowledge engineer must help the expert map into the system.

Moreover, these frameworks are richly intertwined and not neatly separated into distinct subspecialty areas. Woven into the facts and relations are many examples, exceptions, links to other specialty areas. They-appear to be well indexed, for experts seem to have no difficulty in citing examples for every principle and two exceptions for every example. Finally, an expert's store of knowledge is large. Regardless of how one measures the size of an expert's knowledge base, it is almost a truism to say that the more a problem solver knows, the better its advice will be.

As with the data available for solving interesting problems, the expertise that is available may be redundant, with rich inter-dependencies in the reasoning network. In the case of the expertise, as well, the redundancy can be exploited as protection against being led into blind alleys by spurious data or inappropriate heuristics.

6 CONCLUSIONS

Expert systems represent an important set of applications of Artificial Intelligence to problems of commercial as well as scientific importance. There appear to be three main motivations for building an expert system, apart from research purposes:

- Replication of expertise -- providing many (electronic) copies of an expert's knowledge so it can be consulted even when the expert is not personally available. Geographic distance and retirement are two important reasons for unavailability.

- Union of Expertise -- providing in one place the union of what several different experts know about different specialties. This has been realized to some extent in PROSPECTOR [Reboh81] and CASNET [Weiss7b>] which show the potential benefits of achieving such a superset of knowledge bases.

- Documentation -- providing a clear record of the best knowledge available for handling a specific problem. An important use of this record is for training, although this possibility is just beginning to be exploited. [Brown82, Clancey79].

Rule-based systems are currently the most advanced in their system-building environments and explanation capabilities, and have been used to build many demonstration programs. Most of the programs work on analysis tasks such as mcdical'diagnosis, electronic troubleshooting, or data interpretation. The capability of current systems is difficult to define. It is clear, however, that they are specialists in very narrow areas and have very limited (but not totally missing) abilities to acquire new knowledge or explain their reasoning.

One of the important concepts of this style of programming is the *throwaway program.* The existence of a framework system in which to construct a new program allows the expert and knowledge enginccr to focus on *the knowledge* needed for problem solving. Without a framework, they spend more timc on syntactic considerations than on semantic ones. Because the framework is already in place, however, they can readily scrap bad conceptualizations of the problcm solving knowledge. For programs that are built incrementally, as expert systems are, throwing away false starts is important.

Technological innovations will be incorporated into expert systems as the conceptual difficulties of representation and inference in complex domains yield to standardized techniques. Thcse will be most noticeable in the size of the computer and in the input/output of the system. A portable device for troubleshooting, with voice I/O, for example, is not out of the qucstion in the near future.

Systems will use much larger knowledge bases than the few hundred to few thousand rules now used. They will be linked electronically to large data bases to facilitate inference and avoid asking questions whose answers are matters of record. "Smart" interpretation systems will be directly linked to data collection devices, as PUFF is linked to a spirometer, to avoid asking about the data for the case at hand.

For the time being, the major difficult) in constructing' an expert system will remain engineering the knowledge that experts use into a form that is usable by the system. Every problem area and every expert is unique. Nevertheless, many common features have been identified and built into knowlcdge acquisition packages of the major frameworks. Future systems will integrate several modes of knowledge acquisition: some rules can bc extracted from an expert, some from large data bases, and some from experience.

Finally, more powerful system-building frameworks will be developed to reduce the time it takes to iterate on the build-test-refine cycle and to increase the range of applications. There is considerable research in AI of interest to dcsigncrs of expert systcms [Buchanan81], much of it relating to the two ccntral issues of representation and infcrence, some of it rclating to improving the interface bctwccn systcm builders and the emerging system. As this work is integrated into more powerful frameworks, the breadth and depth of applications will increase.

7 ACKNOWLEDGEMENTS . .

This work was supported in part, by research grants to Stanford from ARPA (Contract # MDA903-80-C-0107), NIH (Grant # NIH RR 00785-09), ONR (Contract # N00014-79-C-0302), and by the Fairchild Laboratory for Artificial Intelligence Research.

We received valuable suggestions on earlier drafts from James Bennett, Robert Engelmore, and Michael Genesereth.

REFERENCES

GENERAL READING:

[Infotech81]
--, *Machine Intelligence,* Infotech State of the Art Report Series 9, No. 3. Maidenhead, England: Pergamon Infotech Ltd., 1981.

[Davis76]
Davis, R. and King, J. "An Overview of Production Systems". In E.W. Elcock and D.Michie (eds.), *Machine Intelligence 8.* New York: Wiley, 1976, pp. 300-332.

[Feigenbaum79]
Feigenbaum, E.A. "Themes and Case Studies of Knowledge Engineering", in [Michie79].

REFERENCES CITED:

[Addis80]
Addis, T.R. "Towards an 'Expert' Diagnostic System", *ICL Technical Jnl*, May, 1980, pp.79-105.

[Aikins80]
Aikins, J.S. "Prototypes and Production Rules: A Knowledge Representation for Computer Consultation". *Ph.D. Dissertation, Stanford Univ. Computer Science Department.* STAN-CS-80-814, (1980).

[Amarel81]
Amarel, S. "Problems of Reprcscntation in Heuristic Problem Solving: Related Issues in the Development of Expert Systems", Tech. Report # CBM-TR-118, Laboratory for Computer Science Research, Rutgers Univ., (1981).

[Anderson76]
Anderson, R.H. and Gillogly, J.J. "The Rand Intelligent Terminal (RITA) as a Network Access Aid". *AFIP Proc. 45*, pp. 501-509 (1976).

[Barnett81]
Barnett, J. A., "Computational Methods for a Mathematical Theory of Evidence," *Proc. IJCAI-79,* pp. 868-875 (Univ. of British Columbia, Vancouver, B.C., August 1981).

[Barnett82]
Barnett, J.A., and Erman, L., "Making Control Decisions in an Expert System is a Problem-Solving Task", USC/ISI Tech. Report (April 1982).

[Barr81]
Barr, A. and Feigenbaum, E.A. (eds.) *The Handbook of Artificial Intelligence, vol. I.* Los Altos, CA: Wm.Kaufmann, 1981.

[Barstow79a]
Barstow, D. "An Experiment in Knowledge-Based Automatic Programming". *Artificial Intelligence 12*, pp. 73-119, (1979).

[Barstow79b]
Barstow, D. "Knowledge Engineering in Nuclear Physics". *Proc. IJCAI-79* pp. 34-36, (1979).

[Basden82]
Basden, A., Kelly, B.A., "An Application of Expert Systems Techniques in Materials Engineering", *Proc. of Colloquium on 'Application of Knowledge Based (or Expert) Systems'*, London, (January 1982).

[Bennett79]
Bennett, J. & Engelmore, R. "SACON: A Knowledge-Based Consultant for Structural Analysis". *Proc. IJCAI-79*, pp. 47-49 (1979).

[Bennett81a]
Bennett, J.S. "On the Structure of the Acquisition Process for Rule Based Systems", in [Infotech81].

[Bennett81b]
Bennett, J.S. and Hollander, Clifford R. "DART: An Expert System for Computer Fault Diagnosis", *Proc. IJCAI-81*, pp. 843-845, (1981).

[Bobrow77]
Bobrow, D., and Winograd, T. "An Overview of KRL, a Knowledge Representation Language", *Cognitive Science I:* 1 pp. 3-46 (1977).

[Bonnet81]
Bonnet, A. "Applications de l'Intelligence Artificielle: Les Systemes Experts". *RAIRO Informatique/ Computer Science 15* (4), *325-341, (1981).*

[Brachman77]
Brachman, R.J. "What's In a Concept: Structural Foundations for Semantic Networks". *Int. Jnl Man-Machine Studies 9,* pp. 127-152 (1971).

[Brachman78]
Brachman, R.J., "A STructural Paradigm for Representing Knowledge", Tech. Report *No.* 3605, Bolt Bcranek and Newman, Cambridge, MA, (May 1978).

[Brachman80]
Brachman, R.J. and Smith, B. "Special Issue on Knowledge Representation", *SIGART* 50, (1980).

[Brooks81]
Brooks, R., "A Comparison Among Four Packages for Knowledge-Based Systems", *Proc. of Int. Conf. on Cybernetics and Society,* pp. 279-283 (1981).

[Brown78]

Brown, J.S. and Burton, R., '*Diagnostic Models fo Procedural Bugs in Basic Mathematical Skills", *Cognitive Science 2:2*, pp. 155-192 (1978).

[Brown82]

Brown, J.S., Burton, R.R. & deKleer, J. "Knowledge Engineering and Pedagogical Techniques in SOPHIE I, II and III". In D. Sleeman and J.S. Brown (eds.) *Intelligent Tutoring Systems*. London: Academic Press, 1982.

[Buchanan78]

Buchanan, B.G., and Feigenbaum, E.A. "DENDRAL and Meta-DENDRAL: Their Applications Dimension*'. *Artificial Intelligence 11*, pp. 5-24 (1978).

[Buchanan81]

Buchanan, B.G. "Research on Expert Systems". In D.Michie (cd.) *Machine Intelligence IO,* forthcoming. Also Stanford Univ. Computer Science Dept. Tech Report HPP-81-1 (1981).

[Bundy79]

Bundy, A., Byrd, L., Luger, G., Mellish, C. & Palmer, M. "Solving Mechanics Problems Using Meta-Level Inference", in [Michie79].

[Carhart79]

Carhart, R.E. "CONGEN: An Expert System Aiding the Structural Chemist", in [Michie79].

[Chandrasekaran80]

Chandrasekaran, B., Mittal, S., and Smith, J.W., "RADEX--towards a computer-based radiology consultant". In Gelsema and Kanal (eds.), *Pattern Recognition in Practice,* pp.463-474, North Holland, 1980.

[Clancey79]

Clancey, W.J. "Tutoring Rules for Guiding a Case Method Dialogue". *Intl. Jnl. Man-Machine Studies II,* pp. 25-49 (1979).

[Clancey81]

Clancey, W.J. and Letsinger, Reed "NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching", *Proc. IJCAI-81*, pp. 829-836, (1981).

[Cohen82]

Cohen, P. and Feigenbaum, E.A. (cds.). *The Handbook of Artificial Intelligence, vols. II-III.* Los Altos, CA: Wm.Kaufmann, 1982.

[Davis76]

Davis, R. and King, J. "An Overview of Production Systems". In E.W. Elcock and D.Michie (cds.), *Machine Intelligence 8. New* York: Wiley, 1976, pp. 300-332.

[Davis77a]

Davis, R. and Buchanan, B.G. "Meta-level Knowledge: Overview and Applications, *Proc. IJCAI- 77,* pp. 920-928, (1977).

[Davis77b]

Davis, R. and Buchanan, B.G., and Shortliffe, E.H., "Production Rules as a Representation of a Knowledge-Based Consultation Program", *Artificial Intelligence 8,* pp. 15-45, (1977).

[Davis79]

Davis, R., "Interactive Transfer of Expertise: Acquisition of New Inference Rules", *Artificial Intelligence 12,* pp. 121-157, (1979).

[Davis81]

Davis, R., Austin, H., Carlbom, I., Frawley, B., Pruchnik, P., Sneiderman, R., Gilbreth, JA., "The DIPMETER ADVISOR: Interpretation of Geologic Signals", *Proc. IJCAI-82,* pp. 846-849, (198 1).

[Davis82a]

Davis, R., "Expert Systems: Where Are We? and Where Do We Go From Here?. *The AI Magazine,* 3(2): *3-22, (1982).*

[Davis 82b]

Davis, R., et al, "Diagnosis Based on Description of Structure and Function", *Proc. Second Natl. Conf. on AI",* Pittsburgh, (1982).

[deKleer77]

deKleer, J. Doyle, J., Steele, G. and Sussman, G., "AMORD: Explicit Control of Reasoning", *Proc. Symposium on Artificial Intelligence and Programming Languages,* SIGPLAN Notices, Vol. 12, and SIGART Newsletter No. 64, pp. 116-125 (August 1977).

[deKleer80]

deKleer, J. and Sussman, G.L. "Propagation of Constraints Applied to Circuit Synthesis". *Circuit Theory and Applications 8* (1980).

[Dincbas80]

Dincbas, M., "A Knowledge-Based Expert System for Automatic Analysis and Synthesis in CAD", *Proc. of IFIP Congress* pp. 705-710, (1980).

[Duda76]

Duda, R.O., Hart, P.E., and Nilsson, N.J., "Subjective Bayesian Methods for Rule-Based Inference Systems," *Proc. 1976 Natl. Computer Conf.,* pp. 1075-1082 (AFIPS Vol. 45, 1976).

[Duda78]

Duda, R.O., et al. "Semantic Network Representations in Rule-Based Inference Systems", in [Waterman78].

[Duda79]

R. Duda, J. Gaschnig and P. Hart, "Model Design in the Prospector Consultant System for Mineral Exploration," in [Michie79].

[Duda81]

Duda, R.O., Gaschnig, J.G., "Knowledge-Based Expert Systems Come of Age", *Byte 6: 9*, pp. 238-81, (September, 1981).

[Engelman79]

Engelman,C., Berg, C.H., Bischoff, M., "KNOBS: An Experimental Knowledge Based Tactical Air Mission Planning System" and "A Rule Based Aircraft Identification Simulation Facility", *Proc. IJCAI-79* pp. 247-249, (1979).

[Englemore79]

Engelmore, R.S. & Terry, A. "Structure and Function of the CRYSALIS System". *Proc. IJCAI-79*, pp. 250-256, (1979).

[Ennis82]

Ennis, S.P. "Expert Systems: A User's Perspective of Some Current Tools", *Proc. Second Annual Natl. Conf. on Artificial Intelligence.* Pittsburgh (1982).

[Erman81]

Erman, L.D., London, P.E. & Fickas, S.F. "The Design and an Example Use of HEARSAY-III". *Proc. IJCAI-81*. pp. 409-415, (1981).

[Fagan80]

Fagan, L. "VM: Representing Time-Dependent Relations in a Clinical Setting". *PhD Dissertation, Stanford Univ. Computer Science Department, (1980).*

[Fain81]

Fain, J., Hayes-Roth, F., Sowizral, H., and Waterman, D., "Programming Examples in ROSIE", Rand Corporation, Technical Report N-1646-ARPA (1981).

[Feigenbaum77]

Feigenbaum, E.A. "The Art of Artificial Intelligence: Themes and Case Studies in Knowledge Engineering". *Proc. IJCAI-77*, pp. 1014- 1029, (1977).

[Feigenbaum79]

Feigenbaum, E.A. "Themes and Case Studies of Knowledge Engineering", in [Michie79].

[Fine73]

Fine, T., *Theories of Probability.* New York: Academic Press, 1973.

[Forgy77]

Forgy, C. & McDermott, J. "OPS, A Domain-Independent Production System Language". *Proc. IJCAI-77*, pp. 933-939, (1977).

[Fox82]

Fox, M.S., Allen, B., & Strohm, G. "Job-Shop Scheduling: An Investigation in Constraint-Directed Reasoning". *Proc. Second Annual Natl. Conf. on Artificial Intelligence.* Pittsburgh *(1982).*

[Friedland79]

Friedland, P. "Knowledge-Based Experiment Design in Molecular Genetics". *PhD Dissertation, Stanford Univ., Computer Science Department.* STAN-CS-79-771, (1979).

[Gelernter77]

Gelernter, H.L., Sanders, A.F., Larsen, D.L., Agarival, K.K., Boivie, R.H., Spritzer, G.A., and Searlcman, J.E. "Empirical Explorations of SYNCHEM". *Science 197*, pp. 1041-1049, (1977).

[Genesereth78]

Genesercth, M.R. "Automated Consultation for Complex Computer Systems". *Ph.D. Dissertation, Harvard Univ.,* (September,1978).

[Genesereth81a]

Genesereth, M.R. "The Architecture of a Multiple Representation System", Memo HPP-81-6, Computer Science Dept., Stanford Univ., (May, 1981).

[Genesereth81b]

Genesereth, M., "The Use of Hierarchical Models in the Automated Diagnosis of Computer Systems", Stanford Memo HPP-81-20 (1981).

[Georgeff82]

Georgeff, M.P., "Procedural Control in Production Systems", *Artificial Intelligence 18*, pp. 175-201, (March 1982).

[Goldstein79]

Goldstein, I.P. & Roberts B. "Using Frames in Scheduling". In P. Winston and D. Brown (eds.) *Artificial Intelligence: An MIT Perspective,* vol.1. Cambridge: MIT Press, 1979.

[Gorry78]

Gorry, G.A., Silverman, H. and Pauker S.G. "Capturing Clinical Expertise: A Computer Program that Considers Clinical Responses to Digitalis". *Am Jnl Medicine 64,* pp.452-460, (1978).

[Greiner80]

Greiner, R., and Lenat, D., "A Representation Language's Language". *Proc. First Natl. Conf. on Artificial Intelligence,* Stanford, CA, (August 1980).

[Hayes-Roth78]

Hayes-Roth, F., Watcrnian, D.A. & Lenat, D.B. "Principles of Pattern- Directed Inference Systems". In [Waterman78].

[Heiser78]

Heiser, J.F., Brooks, R.E. and Ballard, J.P. "Progress Report: A Computerized Psychopharmacology Advisor". *Proc. 11 th Collegium Internationale Neuro-Psychopharmacologicum,* Vienna (1978).

[Hewitt72]

Hewitt, C.,' "Description and Theoretical Analysis [using Schemata] of PLANNER:. A Language for Proving Theorems and Manipulating Models in a Robot", *PhD Thesis, Department of Mathematics, M. I.T. (1972).*

[Hewitt75]

Hewitt, C., Smith, B., "Towards a Programming Apprentice", *IEEE Trans Software Eng 1*:1, pp. 26-45 (March 1975).

[Hollander79]

Hollander, Clifford R. & Reinstein, Harry C. "A Knowledge-Based Application Definition System". *Proc. IJCAI-79,* pp. 397-399, (1979).

[Horn81]

Horn, W., Buchstallcr, W., and Trapp, R. "Knowledge Structure Definition for an Expert System in Primary Medical Care", *Proc. IJCAI-1981,* pp. 850-852, (1981).

[Infotech81]

--, *Machine Intelligence*, Infotech State of the Art Report Series 9, No. 3. Maidenhead, England: Pergamon Infotech Ltd., 1981.

[Ishizuka81]

Ishizuka, M., Fu, K-S., and Yao, J.T.P., "Inexact Inference for Rule-Based Damage Assessment of Existing Structures", *Proc. IJCAI-81* pp. 837-842, (1981).

[Kaihara78]

Kaihara, S., Koyama, T., Minamikawa, T., Yasaka, T., "A Rule-Based Physicians' Consultation System for Cardiovascular Diseases", *Proc. of Int. Conf. on Cybernetics and Society", pp. 85-88, (1978.*

[Kandt81]

Kandt, R.K., Newlon, R., "Self-improving Diagnostics for Automatic Testing Equipment", *Proc. of Eighth Semi-annual Seminar/Exhibit*, Pasadena, CA (1981).

[Krueger81]

Krueger, M.W., Culling ford, R.E., Bellavance, D.A., "Control Issues in a Multiprocess Computer-Aided Design System Containing Expert Knowledge", *Proc. Int. Conf. on Cybernetics and Society,* pp. 139-143, (October 1981).

[Kunz78]

Kunz, J.C., Fallat, R.J., McClung, D.H., Osborn, J.J., Votteri, B.A, Nii, H.P., Aikins, J.S., Fagan, L.M. & Fcigenbaum, E.A. "A Physiological Rule-Based System for Interpreting Pulmonary function Test Results." *Tech Report HPP-78-19*, Computer Science Dept., Stanford Univ. (1978).

[Larkin80]

Larkin, J., McDermott, J., Simon, D.P., and Simon, H.A. "Expert and Novice Performance in Solving Physics Problems". *Science 208*:20 (1980).

[Lauriere78]
Lauriere, J.L. "A Language and a Program for Stating and Solving Combinatorial Problems". *Artificial Intelligence 10*, pp. 29-127, (1978).

[Lindberg81]
Lindberg, D.A.B., Gaston, L.W., Kingsland, L.C., Vanker, A.D., "AI/COAG, A Knowledge-Based System for Consultation About Human Hemostasis Disorders:Progress Report", *Proc. of the Fifth Annual Symposium on Computer Applications in Medical Care,* pp. 253-257, (1981).

[Lindsay80]
Lindsay, R., Buchanan, B.G., Feigenbaum, E.A., and Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project.* New York: McGraw Hill, 1980.

[Lusted68]
Lusted, L.B., "Introduction to Medical Decision Making", Charles C. Thomas Co., Springfield, Illinois, (1968).

[Mamdani82]
Mamdani, E.H., "Rule-Based Methods for Designing Industrial Process Controllers", *Proc. of Colloquium on 'Application of Knowledge Based (or Expert) Systems',* London, (January 1982).

[McColl79]
McColl, D.C., Morris, P.H., Kibler, D.F., and Bechtel, R.J. "STAMMER2 Production System for Tactical Situation Assessment", Tech. Report TD 298, Naval Ocean Systems Center, San Diego, CA (Oct., 1979).

[McCune77]
McCune, B.P., "The PSI Program Model Builder Synthesis of Very High-Level Programs", *Sigplan Not.* Vol. 12, No. 8, pp. 130-139 (1977).

[McDermott80]
McDermott, J., "R1: An Expert in the Computer Systems Domain", *Proc. First Annual Natl. Conf. on Artificial Intelligence,* pp. 269-271, Stanford Univ., Stanford, CA, (1980).

[McDermott81]
McDermott, J., "XSEL: a computer salesperson's assistant", In J. Hayes and D. Michie (eds.) *Machine Intelligence 10*, forthcoming.

[Michie79]
Michie, Donald. *Expert Systems in the Micro-Electronic Age.* Edinburgh: Edinburgh Univ. Press, (1979).

[Michie80]
Michie, Donald. "Expert Systems". *The Computer Journal* 23(4):369-376, *(1980).*

[Minsky75]
Minsky, M. "A Framework for Representing Knowledge". In P. Winston (ed.) *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975.

[Mitchell79]

Mitchell, T.M. "Version Spaces: An Approach to Concept Learning". *PhD Dissertation, Technical Report HPP- 79-2, Computer Science Department,* Stanford Univ. (1979).

[Moses71]

Moses, J. "Symbolic Integration: The Stormy Decade". *Communications ACM 8,* pp. 548-560, (1971).

[Newell72]

Newell, A. & Simon, H.A. *Human Problem Solving.* New York: Prentice-Hall, 1972.

[Newell73]

Newell, A. "Production Systems: Models of Control Structures", in W.Chase (ed.) . *Visual Information Processing.* New York: Academic Press, 1973.

[Newell76]

Newell, A. & Simon, H.A. "Computer Science as Empirical Inquiry: Symbols and Search". The 1976 ACM Turing Lecture. *Communications ACM* 19, pp. 113-126 (1976).

[Nii79]

Nii, H.P. and Aiello, N. "AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs". *Proc. IJCAI-79*, pp. 645-655, (1979).

[Nii82]

Nii, H.P., Feigenbaum, E.A. , Anton, J.J. & Rockmore, A.J. "Signal-to-Symbol Transformation: HASP/SIAP Case Study". *The A I Magazine 3:2* pp.23-35, (1982).

[Nilsson80]

Nilsson, N.J., *Principles of Artificial Intelligence,* Palo Alto, CA: Tioga Press, 1980.

[Novak80]

Novak, G., Araya, A.A., "Research on Expert Problem Solving in Physics", *Proc. first Natl. Conf. on Artificial Intelligence,* Stanford, CA, (August 1980), pp. 178-180.

[O'Shea79]

O'Shea, T., "Rule-Based Computer Tutors", *Proc. of 1979 AISB Summer School*, pp. 226-232, (1979).

[Patil81]

Patil, Ramesh S., Szolovits, P., Schwartz, Wm. B., "Causal Understanding of Patient Illness in Medical Diagnosis", *Proc. IJCAI-81*, pp. 893-899, (1981).

[Pednault81]

Pednault, E.P.D., Zucker, S.W., and Muresan, L.V. "On the Independence Assumption Underlying Subjective Bayesian Updating", *Artificial Intelligence 16*, pp. 213-222, (May 1981).

[Pinson81]

Pinson, S. "Reprcscntation dcs Connaissanccs dans lcs Systcmes Experts." *RAIRA Informatique/ Computer Science 15:4*, pp. 343-367, (1981).

[Pople77]

Pople, H.E. "The Formation of Composite Hypotheses in Diagnostic Problem Solving--an Exercise in Synthetic Reasoning". *Proc. IJCAI-77,* pp. 1030-1037, (1977).

[Reboh81]

Reboh, R., "Knowledge Engineering Techniques and Tools in the Prospector Enviranment", Tech. Note 243, Artificial Intelligence Center, SRI International, Menlo Park, CA (June, 1981).

[Reggia78]

Reggia, J.A. "A Production Rule System for Neurological Localization". *Proc. 2nd Annual Symposium on Computers Applied to Medical Care,* pp. 254-260, (1978).

[Reinstein81]

Reinstein, H.C., Aikins, J.S., "Application Design: Issues in Expert System Architecture", *Proc. IJCAI-81*, pp. 888-892, (1981).

[Rivlin80]

Rivlin,J.M. Hsu, M.D., Marcal, P.V., "Knowledge Based Consultation for Finite Element Structural Analysis", *Tech. Report, MARC Analysis Research Corp.* Palo Alto, CA (1980).

[Roberts771

Roberts, R.B. & Goldstein, I.P. *The FRL Primer.* MIT AI Lab Memo #408, (1977).

[Savage71]

Savage, L.J., "Elicitation of Personal Probabilities and Expectations" *Jnl of Am Statistical Assn,* pp. 783-801, (December 1971).

[Scott77]

Scott, A.C., Clancey, W., Davis, R., and Shortliffc, E.H. "Explanation Capabilities of Knowledge-Based Production Systems". *American Jnl of Computational Linguistics.* Microfiche 62, (1977).

[Shafer76]

Shafer, G., *A Mathematical Theory of Evidence,* Princeton, NJ: Princeton Univ. Press, 1976.

[Shortliffe75]

Shortliffe, E.H., and Buchanan, B.G., "A Model of Inexact Reasoning in Medicine", *Mathematical Biosciences 23,* pp. 35 I-379 (1975).

[Shortliffe76]

Shortliffe, E.H. *Computer Based Medical Consultations: MYCIN.* New York: American Elsevier, 1976.

[Shortliffe81]

Shortliffc, E.H., Scott, A.C., Bischoff, M.B., Campbell, A.B., van Melle, Wm., Jacobs, C.D., "ONCOCIN: An Expert System for Oncology Protocol Management", *Proc. IJCAI-81*, pp. 876-881, (1981).

[Sridharan80]

Sridharan, N.S. "Representational Facilities of AIMDS: A Sampling". *Tech. Report # CBM-TM-86,* Dept. of Computer Science, Rutgers Univ., 1980.

[Stallman77]

Stallman, R.M., Sussman, G.J., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis", *Artificial Intelligence 9,* pp.135- 196 (1977).

[Stefik78]

Stefik, M. "Inferring DNA Structures from Segmentation Data". *Artificial Intelligence 11*, pp. 85-114, (1978).

[Stefik79]

Stefik, M. "An Examination of a Frame-Structured Representation System". *Proc. IJCAI-79,* pp. 845-852, (1979).

[S tefik80]

Stefik, M. *Planning with Constraints.* Ph.D. Dissertation, Stanford Univ. Computer Science Department. STAN-CS-80-784, (1980).

[S tefik82]

Stefik, M. et al. "The Organization of Expert Systems, A Tutorial", *Artificial Intelligence I8* pp. 135-173 (1982).

[Sussman78]

Sussman, G.A. "SLICES: At the Boundary Between Analysis and Synthesis". In J.C. Latombe (ed.) *Artificial Intelligence and Pattern Recognition in Computer -Aided Design.* New York: North-Holland, 1978.

[Swartout77]

Swartout, W. "A Digitalis Therapy Advisor with Explanations", *Proc. IJCAI-77,* pp. 819-823, (August 1977).

[Swartout81]

Swartout, W. "Explaining and Justifying Expert Consulting Programs". *Proc. IJCAI-81*, pp. 815-822, (1981).

[Szolovits77]

Szolovits, P., Hawkinson, L.B., and Martin, W.A. "An Overview of OWL, a Language for Knowledge Representation". *MIT LCS TM 86, (1977).*

[Szolovits78] .

Szolovits, P., Paukcr, S.G., "Categorical and Probabilistic Reasoning in Medical Diagnosis", *Artificial Intelligence I1*, pp. 115-l 44, (1978).

[Tsotsos81]

Tsotsos, J.K., "On Classifying Time-Varying Events", *IEEE Computer Society Conf. on Pattern Recognition and Image Processing,* pp. 193-199 Toronto (198 1)

[vanMelle80]

van Mellc, W. "A Domain Independent System that Aids in Constructing Knowledge Based Consultation Programs". *PhD Dissertation, Stanford Univ. Computer Science Department.* STAN-CS-80-820, (1980).

[Warren77]

Warren,, D., et al. "PROLOG: The Language and Its Implementation Compared with LISP". *Proc. SIGART/SIGPLAN Symposium on Programming Languages,* Rochester, NY, 1977.

[Waterman78]

Waterman, D.A. and Hayes-Roth, F. (eds.). *Pattern Directed Inference Systems. New* York: Academic Press, 1978.

[Waterman79]

Waterman, D.A. "User-Oriented Systems for Capturing Expertise: A Rule-Based Approach", in [Michie79].

[Weiss79a]

Weiss, S. and Kulikowski, C. "EXPERT: A System for Developing Consultation Models". *Proc. IJCAI-79,* pp. 942-947, (1979).

[Weiss79b]

Weiss, S., Kulikowski, C., Amarel, S., and Safir, A. "A Model-Based Method for Computer-Aided Medical Decision Making". *Artificial Intelligence II,* pp. 145-172, (1979).

[Weyrauch80]

Weyhrauch, R.W. "Prolcgomena to a Theory of Mechanized Formal Reasoning", *Artificial Intelligence* 13: l-2 (April, 1980).

[Wipke77]

Wipke, W.T., Braun, H., Smith, G.,Choplin, F., and Sieber, W. "SECS--Simulation and Evaluation of Chemical Synthesis: Strategy and Planning". In W.T. Wipke and W. J. House (eds.), Computer Assisted Organic Synthesis. Washington, D.C.: *American Chemical Society,* pp. 97-127, (1977).

[Yu79]

Yu, V.L., Fagan, L., Wraith, S.M., Clancey, W.J., Scott, A.C., Hannigan, J., Blum, R., Buchanan, B.G., Cohen, S.N., Davis, R., Aikins, J.S., vanMelle, W., Shortliffe, E.H., and Axline, S. "Antimicrobial Selection for Meningitis by a Computerized Consultant: A Blinded Evaluation by Infectious Disease Experts". *Jnl Am Medical Assn* 241:12, pp. 1279-1282 (1979).

[Zadeh65]

Zadeh, L.A., "Fuzzy *Sets," Information and Control,* pp. 338-353 (1965).

[Zadeh78]

Zadeh, L.A., "Fuzzy Sets as a Basis for a Theory of Possibility", *Fuzzy Sets and Systems,* Amsterdam: North-Holland 1978.

APPENDIX A.
Answers to User's Questions about this Consultation and About the Knowledge Base

```
    Please ask any simple-English questions regarding the above
consultation session or the overall knowledge of the consultation
system.  When discussing the consultation above, please refer to
organisms, cultures, etc. by their numeric names (eg. ORGANISM-1).
Questions which were asked during the consultation may be specified
by the numbers that preceded them above, (e.g., 'WHY DID YOU ASK
QUESTION 1?').  Control-O will abort type-out, or 'cancel' the current
question.  Type HELP for a list of the options available.  Enter the
word STOP when you have no further questions.

Type HELP for the list of question options.


** WHY DID YOU TREAT E.COLI?

ITEM-1 is E.coli associated with meningitis.  I decided to treat ITEM-1 on
     the basis of clinical evidence only.

The following rules concluded about:
     the likelihood that e.coli is one of the organisms (other than those
          seen on cultures or smears) which might be causing the infection

RULE545 was used to conclude that e.coli is one of the organisms (other than
     those seen on cultures or smears) which might be causing the infection
     (.74).  The last question asked before the conclusion was made was 35.
RULE557 was used to conclude that e.coli is one of the organisms (other than
     those seen on cultures or smears) which might be causing the infection
     (.93).  The last question asked before the conclusion was made was 37.

The parameters most frequently used in these rules were:
     the infection which requires therapy
     whether a smear of the culture was examined
     whether organisms were seen on the stain of the culture
     the type of the infection

Would you like a summary of ALL the parameters used by these rules?  ** YES

          Parameter     (rules with this parameter)

the white count from the patient's peripheral CBC (in thousands) (557)
whether the results of a recent CBC of the patient are available (557)
whether the infection was acquired while the patient was hospitalized (545)
the age of the patient (545)
the relevant items from the history of the patient (545)
the type of the infection (545, 557)
whether organisms were seen on the stain of the culture (545, 557)
whether a smear of the culture was examined (545, 557)
the infection which requires therapy (545, 557)
```

**WHY DIDN'T YOU GIVE TETRACYCLINE FOR E.COLI IN REC-1?
Shall I describe the therapy algorithm for you?** YES .
DESCRIPTION OF THERAPY SELECTION
[Type ↑O to abort printout]

MYCIN is designed to provide optimal therapy recommendations. The program uses the following three steps:
The first step ranks the drugs for each organism that requires therapy. Rules for doing this take into account invitro (lab) sensitivities and the patient's response to prior and current therapies. In the absence of other information, default lab sensitivities are used. Preference is given to current therapies if certain criteria are satisfied.
Next, 1 or 2 drugs are selected to cover for the organisms which are most likely to be causing the infection. As many of the "less likely" organisms as possible are treated, but they are disregard altogether if 3 or more drugs are necessary. Throughout, preference is given to drugs that appear higher in the treatment lists. (These lists can be examined via the RX option.)
Finally, the set of drugs is tested for patient factors that preclude their administration, such as allergies, age, and renal function.
TETRACYCLINE was not prescribed for ITEM-1
TETRACYCLINE was not prescribed for ITEM-1 in RECOMMENDATION-1 because:

TETRACYCLINE was discounted for ITEM-1 because there is evidence that this e.coli is not sensitive to it.

Would you like to see some details?** NO

** HOW DO YOU TREAT PNEUMOCOCCAL PNEUMONIA
The MYCIN system currently can treat the following infections: BACTEREMIA MENINGITIS CYSTITIS. Because the knowledge base is incomplete, all other infections will be treated like bacteremia.

For treatment of DIPLOCOCCUS-PNEUMONIAE in BACTEREMIA
The drugs of choice (ranked by relative efficacy, assuming sensitivity) are:
1st choice:
PENICILLIN
AMPICILLIN
2nd choice:
NAFCILLIN
CEPHALOTHIN
ERYTHROMYCIN
CLINDAMYCIN
3rd choice:
TETRACYCLINE
CHLORAMPHENICOL
VANCOMYCIN .
Duration of therapy: 28 days

** WHAT DOSE OF ERYTHROMYCIN DO YOU RECOMMEND?
When ERYTHROMYCIN is given for BACTEREMIA, the recommended dosage is:
if age is > 12 years then 12 mg/kg q6h IV
     else if age is > .08 years then 5 mg/kg q6h IV
     else if age is > .02 years then 5 mg/kg q12h IV
[Different dosages are recommended for different infections.]

** HOW DO YOU TREAT CYSTITIS
In general, treatment for an infection depends on the causative organism.
Please include in your question the organism of interest.

** STOP