PN-AAZ-367
54125

Working Paper Series

Working Paper No. 7
"Expert Systems, Microcomputers and
Operations Research"
by
Arjang A. Assad and Bruce L. Golden
College of Business and Management
University of Maryland at College Park

May, 1987

Fisheries Stock Assessment
Title XII
Collaborative Research Support Program

The Fisheries Stock Assessment CRSP (sponsored in part by USAID Grant No. DAN-4146-G-SS-5071-00) is intended to support collaborative research between U.S. and developing countries' universities and institutions on fisheries stock assessment and management strategies.

This Working Paper has been produced by the collaborative efforts of The University of Maryland-Center for Environmental and Estuarine Studies and The University of Costa Rica-Centro de Investigacion en Ciencias del Mar y Limnologia (CIMAR) in association with the University of Delaware and the University of Miami. Additional copies are available from the CRSP Management Office and from:

Chesapeake Biological Laboratory
Center for Environmental & Estuarine Studies
The University of Maryland
P.O. Box 38
Solomons, Maryland 20688

EXPERT SYSTEMS, MICROCOMPUTERS, AND OPERATIONS RESEARCH

by

Arjang A. Assad and Bruce L. Golden

College of Business and Management
University of Maryland at College Park, MD 20742

Abstract

The goal of this article is to review microcomputer-based software for building expert systems. There are now at least 15 commercially-available tools to facilitate the development of expert systems on microcomputers. First, the field of expert systems is introduced in general terms. Then, issues relating more directly to microcomputers are studied in detail. Trends and future directions are also discussed.

1. INTRODUCTION

Simply defined, an expert system "is a computer program that has built into it the knowledge and capability that will allow it to operate at the expert's level" [14]. Such a program should exhibit "a high level of performance on problems that are difficult enough to require significant human expertise for their solution" [12]. Application areas for expert systems include medical diagnosis, evaluation of geological sites, identification of drilling problems, design of computer hardware, and the elucidation of the chemical structure of certain compounds.

The goal of this article is to review microcomputer-based software for building expert systems. There are now nearly 20 commercial software tools for the development of expert systems on microcomputers. The capabilities of these packages, called expert system shells, are examined in this paper. Since a discussion of expert system shells presupposes an understanding of the terminology, structure, and functional features of expert systems, the first part of the article provides a general discussion of the issues surrounding expert systems that should serve two purposes. First, it establishes the key terminology used in discussing expert systems. Sections 2 and 3 of this paper aim to perform this task.

Second, by considering the past development of expert systems, which involved designing customized systems on mainframe machines, one can provide a background for the potential applications of expert systems that one may develop through shells. We believe that such a background is important in assessing the promise of microcomputer-bases expert systems. Much of the current interest in commercial expert systems is a result of "glory by association",

as the advocates of such systems are drawing upon the successes of a few impressive and well-publicized major developmental efforts on mainframes. These successes have raised the market's expectations regarding the potential of expert systems. For this reason, we summarize some of the current issues in the implementation and evaluation of expert systems in Section 4 and present some preliminary thoughts on the relevance of such systems to operations research in Section 5. A review of the current commercial expert system shells forms the main subject of Section 6.

In the last two years, expert systems have received an extraordinary level of attention in computer-related periodicals and have been hailed by some as the prime hope of commercial artificial intelligence (AI). As we just mentioned, part of the enthusiasm regarding the potential of expert systems derives from impressive reports of the capabilities of several well-known systems. For example, medical diagnosis systems are reported to surpass expert physicians in certain cases. XCON, an expert system for configuring VAX computers, is reported as saving the vendors around $200,000 per month [24]. PROSPECTOR, a system for exploratory geology, is credited with the discovery of an ore deposit in a region where exploration had been terminated [11]. Chemical structures identified by DENDRAL from mass-spectroscopic data have been published in respected chemistry journals. EURISKO, an expert system with learning capabilities, is advertised as winning the national war game against experienced human players [29]. These "success stories" have naturally generated much curiosity about expert systems applications, in turn giving rise to a variety of courses and seminars in this area.

Various sources have emphasized the large commercial potential of expert systems. One of the earlier popularized accounts has to do with the role of expert systems within Japanese Fifth Generation computers [14]. Some of the leading firms are now investing in in-house artificial intelligence capabilities. One example of the use of such internal research involves the use of expert systems to capture the knowledge of experts prior to their retirement from the company [24]. In the USA, the sales of expert system development tools has been estimated at $20 million in 1984 [26], and the combined size of the AI market is expected to grow to 2-3 billion dollars by 1990 [20]. One authority in this area states that many applications afford annual savings exceeding $10 million [22]. A firm specializing in microcomputer packages that offer advice on human interactions estimates its 1985 sales at $11 million [1].

To balance these optimistic and enthusiastic claims, some have voiced their skepticism as to both the current capabilities and the growth potential of most commercial expert systems [30], [1]. As with other newly-commercialized software, certain reports on expert systems are prone to exaggeration. This places a special burden on the user of such tools to reach an independent judgment as to the utility of expert systems in his work. On a more philosophical level, the article by Stevens in [16] speculates on the negative sociocultural implications of widespread use of expert systems.

While many other examples can be cited, the preceding discussion should indicate that the development of expert systems has generated much enthusiasm as well as some controversy. The lack of concensus

as to what qualifies as an expert system further complicates the matters. A variety of systems with widely different functions are currently called "expert systems" due to the latitude with which this term can be interpreted. The next section, which reviews the function and structure of an expert system, should clarify our use of the term within this paper.

2. FUNCTION AND STRUCTURE OF AN EXPERT SYSTEM

Functionally, expert systems are designed to emulate expert human performance, implying that the user's interaction with the computer program is modeled after his interaction with a human expert. In most applications, the interaction follows the **consultation** paradigm in which the expert's advice is sought to solve a **selection** problem. In a selection problem, we have a prespecified list of alternatives from which we must select one or more items that match the constraints of the problem. Thus, in medical diagnosis, an appropriate disease that matches the patient's symptoms must be selected and in chemical structure analysis, a structure consistent with the experimental data must be found. Other authors have used the terms classification or diagnosis problems in lieu of selection [49].

The term consultation mentioned above refers to the process of interaction with the program. Consider the (minor) problem of selecting a text for a course in operations research. An instructor teaching the course for the first time may consult a more experienced colleague and seek a recommendation. This colleague will ask a number of questions about the level of the course and its prerequisites, the students' backgrounds, the instructor's preferences, etc. He will then combine this knowledge with his own guidelines (or rules) for selecting textbooks to arrive at a recommended text (or texts). If required, he can justify his choice and give a brief description of his line of reasoning. A basic goal of expert systems is to simulate this process in specific and complex domains of knowledge.

For an expert system to consult like a human expert, it must ideally have the following characteristics.

* Its performance should be comparable to the human expert in terms of reliability and accuracy of its solutions.
* It should be able to justify and explain its results.
* The knowledge it draws upon should be easy to expand or update.
* It should be able to generate multiple solutions and qualify its recommendations if necessary.
* It should be able to accept the problem description in lay terms and translate it into its own representation of the problem space.
* It must disregard irrelevant data.
* It must be capable of checking the problem description and inputs for consistency and inacccuracies and provide the user with some measure or indication of the inconsistency.
* Given a problem outside its domain of expertise, it should be able to identify it as such and decline offering a solution.

In addition to these, the consultation process itself should conform to the human model. This means that random queries and lengthy detours in the questioning process should be avoided. Indeed, the system must give the appearance of following a directed questioning strategy that passes from general questions to more specific ones. Moreover, this should be conducted in some reasonable substitute for natural language (say, a subset of the English language) so that the user would not be hampered in his consultation with the system. This is already a non-trivial list of requirements. Certain authors would augment this list with other desirables such as learning, handling exceptions, and restructuring of the knowledge, among others [8], [54].

The design of an expert system is guided by a model of the problem-solving behavior and based on an understanding of the key factors affecting the performance of human experts. A basic assumption of expert system design is that "expert human performance rarely conforms to some rigorous algorithmic process," yet "such performance does lend itself to computerization" [23]. The knowledge is assumed to be representational, meaning that it essentially consists in "representing symbolically facts about the world" [23]. This explains the use of the computer for symbolic (as opposed to numerical) processing in this field.

Another underlying theme is that the expert's knowledge is largely heuristic and not formalized as a rigid procedure. The level of certainty attached to this knowledge may also vary. Much of the effort in designing an expert system is devoted to the extraction of this knowledge and its representation in a form the computer can utilize.

The process of acquiring, formalizing, and utilizing the rules of good practice or plausible reasoning is called knowledge engineering [13]. To accomplish this task, the knowledge engineer, who is already familiar with AI and computer techniques, works closely with the expert to acquire domain-specific knowledge and translate it into computer-useable form. Clearly, this view of human knowledge, which has its roots in the history of AI, is markedly different from the utility maximization view of decision-making that is prevalent in operations research. Correspondingly, the reliance of expert systems on algorithmic procedures appears to be minimal. We now outline the structure of an expert system and the method by which it reaches a conclusion.

Expert System Architecture

An expert system may be viewed as comprising three components--the knowledge base, the inference engine, and the user interface as shown schematically in Figure 1. Briefly put, the knowledge base contains the knowledge extracted from the expert, possibly through a knowledge acquisition subsystem, in a representation suitable for symbolic processing. The inference engine contains the methods for manipulating the knowledge to arrive at conclusions. The user interface consists of tools that aid the interaction with the user such as basic parsing for the language in which the consultation is conducted. An expert system development tool is a program for constructing an expert system. This program includes a structure for the knowledge base, an inference engine, and appropriate user interfaces but no domain-specific knowledge. This last item is provided by the user to build an expert system for a specific application. Such a program is also called an expert system shell, a term that we adopt.

The knowledge base (KB) is at the heart of an expert system. In the oft-cited words of Feigenbaum [12], "the theme is that in the knowledge is the power". This is to say that the power of an expert system derives primarily from the wealth of domain-specific knowledge it can efficiently encode and not from the strength or ingenuity of its reasoning methods. Thus, "expert systems must be knowledge-rich even if they are methods-poor" [13]. For this reason, expert systems and related programs are called knowledge-based systems. However, we should note that in reviewing microcomputer-based tools, our discussion will focus on expert system development tools (or shells) whose knowledge bases have to be constructed by the user.

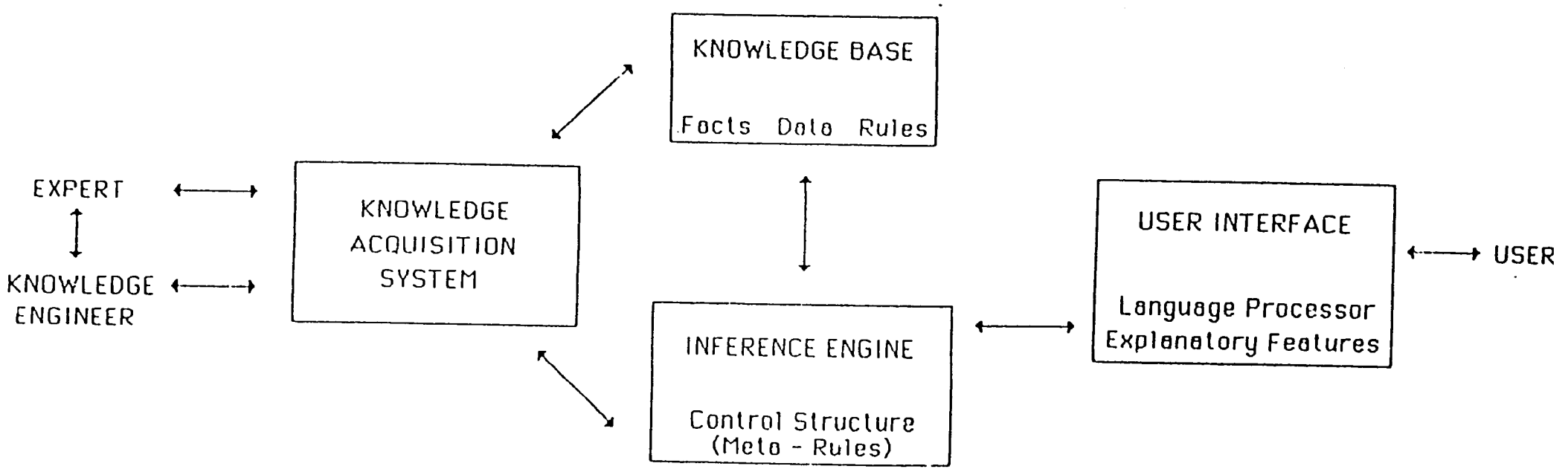

Figure 1. Schematic Structure of a Rule-Based Expert System

A crucial factor in the design of a KB for a particular domain is **knowledge representation**, i.e., how the knowledge is encoded within the program. This issue is discussed in detail in Volume 1 of [4] and also in [37]. We start by focusing on one such representation known as a **production system**. This representation consists of a set of **production rules**, also known as **if-then rules**. Each rule is of the form "If P then Q" for certain expressions P and Q , respectively known as the antecedent (if clause) and the conclusion (then clause), respectively. For example, the rule might read

> "If the course is an MBA course and the level is introductory, then text should contain case studies."

Note that logical connectives (Boolean operators) are allowed in the if clause. Once the condition (if clause) in a rule is found to be satisfied, we say that the rule **fires** and the action in the conclusion can be carried out or stored. Production rules are simply called rules in what follows. A KB using the production rule representation is called a **rule-based system**.

The rules comprising the KB must be extracted from the expert. This requires the generally time-consuming extraction of rules from the expert's informal and implicit reasoning practice. The process of making this knowledge available to the program is called **knowledge acquisition**. The potential difficulty of this process is viewed as a shortcoming of rule-based systems. Nonetheless, if-then rules have certain advantages. First, each rule represents a "chunk" of knowledge that conveys some information by itself. This results in the modularity of the KB and facilitates its enlargement, modification, and refinement. Second, in a variety of domains, rules provide an easy way to represent knowledge. One needs only to think about how commonplace books of rules and

regulations are. Table 1, and some of our later discussion, outlines the advantages and disadvantages of some of the other representation schemes used in expert systems.

In addition to the rules, the KB contains factual knowledge, in the form of facts, that represents the information generated through the dialogue with the user or other assertions of a general nature. For example, in the textbook selection case, the user must specify the values of certain expressions such as the level of the course and its prerequisites. One way of doing this is through attribute-object-value (A-O-V) triplets. These are statements of the form

< Attribute > of <Object> is <Value> , as in

The format of Text #45 is case-oriented.

Another example of a fact is a commonly accepted assertion such as "simulation requires computers".

The Inference Engine

An expert system derives its conclusions by drawing a sequence of inferences from the knowledge available to it in some order. The inference engine contains the control strategy for this process of knowledge utilization. If we view the problem as a search in some state space, the inference engine determines the manner in which the search is performed. The role of AI heuristics is to avoid the combinatorial explosion of the state space and to limit the computational requirements by avoiding blind search. Two well-known control strategies are backward chaining and forward chaining (see [42] or [51]).

Backward chaining is similar to backward recursions in dynamic programming. Starting with a goal expression whose value must

be determined, the engine searches the KB for rules that allow it to conclude a value for the goal. These then serve as the new goals and the search continues. In the process, the system is using rules and facts of the KB and may, of course, also seek the values of certain expressions from the user. This strategy is also called **goal-driven search.** Forward chaining, also known as **data-driven search**, starts with the known values of certain expressions and uses these to infer the values of other expressions until a goal state is reached.

In general, the order in which the rules are examined in such search processes depends on their arrangement or sequence in the KB. A change in this order may alter the search path and its results. Clearly, avoiding dead-ends and circuitous or unpromising search paths directly affects the power and speed of the system, and is at the heart of heuristics design in AI. Nau [37] has noted that both approaches may be written as nondeterministic search procedures, while an implementation of these strategies in LISP may be found in [53].

Backward chaining is found in a number of expert systems since it generates a more directed (and smaller) sequence of questions. Forward chaining is used in expert systems such as DENDRAL that accept information (mass spectroscopic data) directly from measurement instruments. One may, of course, control the search through specially-designed heuristics as described in [23]. For example, Naylor, in chapter 6 of [16], mentions a "sideways chaining" procedure in which the rules are assigned ranks and the one with the highest rank is evaluated next. Roughly, this means that one seeks the information that, if made available, would

induce the highest shift in the known evidence. More generally, it has been recognized that the designer can use rules to influence the control strategy of an expert system. These guidelines, called meta-rules, might direct the search by going to specific portions of the KB or even modify the rules contained therein. The shell M.1 contains some elementary examples of meta-rules for displaying information at certain points of the search or abandoning the search for the value of an expression. However, the use of meta-rules can be significantly more sophisticated and constitutes an area of ongoing research interest. Just as rules make statements about the knowledge domain, meta-rules make statements about the rules and their usefulness to the system that embody knowledge of the expert system itself. Some of the research on meta-rules is reported in chapter 28 of [6].

Interaction With the User

Given the goal of having expert systems emulate the human expert, careful attention must be paid to the interaction between the user and the program. One important component of this interaction is the questioning strategy. The expectations of the user may place certain constraints on this strategy. Most people would feel uncomfortable with an apparently random or aimless string of questions and quickly lose interest in the program. In medical diagnosis, certain questions about the patient are expected to precede others. In some applications, one may want to group subsets of questions so that they are asked together. Beyond these simple constraints, one would want the system to choose the "best next question to ask of the user" . This issue is called question selection by Weiss and Kulikowski [49] who give a number of criteria for this selection. Since the order of the questions depends on the organization of the KB and the

control strategy, the issues of question selection and control strategy are highly interrelated.

Another form of interaction with the user has to do with justification and explanation. This is not simply a courtesy to the user, but also a major factor in the acceptance of an expert system. In rule-based systems, such explanation currently involves a listing of the various rules and facts used in the inference chain leading to the conclusion. The requirement of transparency implies that the program's reasoning should be available to the user (as well as the developer) for easy inspection.

An expert system should also provide easy interaction both in the course of knowledge acquisition and in the questioning process. At the simplest level, the system should check for typographic and syntax errors. In some systems, such as M.1 and TIMM-PC, legal responses to certain questions (or admissible values for expressions) can be defined in advance so that an illegal response would result in an error message. A more advanced capability would involve checking the semantics of the input knowledge, say by examining the consistency of the rules. Naturally the design of an easy-to-use interface assumes even greater importance in microcomputer-based systems. For example, earlier work on large expert systems pointed to the desirability of increasing the expert's direct interaction with the expert system by providing an intelligent editing program, thereby reducing the need for the knowledge engineer as an intermediary [23]. Given the trend in commercial personal computer software towards eliminating the need for a programmer or advanced programming expertise on the part of the user, the preceding objective assumes even

greater importance for expert system shells designed to run on microcomputers.

The basic architecture of an expert system sketched in this section was deliberately kept simple in order to illustrate the basic terminology. The next section describes certain refinements of expert system features as well as other approaches to their design.

3. UNCERTAINTY AND OTHER REFINEMENTS

In this section, we first discuss how uncertainty is incorporated into the inference methodology and then mention a number of design principles for various components of an expert system that may be used as alternatives to those covered in the last section.

Incorporating Uncertainty

An if-then rule was described in deterministic terms in the last section in the sense that the conclusion is certain to follow if the antecedent clause is satisfied. Often, however, the expert might not state the rule so emphatically. Consider the statement "If the course is an MBA course, then the text should be case-oriented with 80% certainty". This rule is of the form "If P then Q (CF)", where CF denotes the certainty factor (CF) associated with the statement. Shortliffe attached values between -1.0 and 1.0 to statements in MYCIN to incorporate uncertainty [43], [44]. Absolute confidence in a statement or its negation corresponds to the value 1.0 or -1.0, respectively. Other implementations have used scales of -100% to 100% or -5 to 5 for the CF's.

The main contribution of Shortliffe [44] was to propose a model of inexact reasoning by constructing rules for propagating the uncertainty through the inference chain. Figure 2 illustrates the computation of certainty factors in the two systems MYCIN and M.1. An updated explanation of the procedure in MYCIN can be found in part 4 of [6] where the reader can find remarks on the history of the subject and an attempt at grounding the approach in a theory of evidence .

Computational Rules for Certainty Factors

(1) Let CF(.,R) denote the Certainty Factor (CF) resulting from the rules listed in R. Given two rules R1 and R2, CF(.,R1,R2) = CF(.,R1) + CF(.,R2)(1 - CF(.,R1))

(2) The CF of a conjunction of statements equals the minimum of the CF's of the individual statements. Schematically,

$$CF(A \text{ AND } B) = Min\{CF(A), CF(B)\}$$

(3) Disjunctions result in the maximum of the individual CF's. Schematically,

$$CF(A \text{ OR } B) = Max\{CF(A), CF(B)\}$$

(4) CF of the conclusion of an uncertain premise = (CF of premise)(CF of conclusion if premise were certain)

Available Information

Statements	P	Q	S	U	V
CF	0.8	0.75	0.9	0.6	0.4

Rule R1: IF P AND Q AND S THEN W (0.8)

Rule R2: IF U OR V THEN W (0.5)

Computation of CF for W in MYCIN

CF(W,R1) = Min(0.8, 0.75, 0.9) x 0.8 = 0.6
by (C2) and (C4)

CF(W,R2) = Max(0.6, 0.4) x 0.5 = 0.3
by (C3) and (C4)

CF(W,R1,R2) = 0.6 + 0.3 (1 - 0.6) = 0.72
by (C1)

Computation of CF for W in M.1

Note: Instead of using (C3), M.1 replaces R2 by the two rules:

R2a: IF U THEN W (0.5)

R2b: IF V THEN W (0.5)

CF(W,R1) = 0.6 as before

CF(W,R2a) = 0.6 x 0.5 = 0.3 by (C4)

CF(W,R2b) = 0.4 x 0.5 = 0.2 by (C4).

CF(W,R1,R2a) = 0.6 + 0.3(1 - 0.6) = 0.72

CF(W,R1,R2a,R2b) = 0.72 + 0.2(1 - 0.72) = 0.776

by repeated application of (C1).

Figure 2. Examples for the Calculation of Certainty Factors

Certainty factors should not be interpreted as probabilities. Research has revealed that one needs highly restrictive assumptions to derive the rules governing CF's from the traditional concepts of probability theory. The ad hoc nature of these rules has led researchers to seek other methods of incorporating uncertainty into an expert system. Using Bayesian theory or Fuzzy Logic are two such alternatives that we now briefly comment on.

In the Bayesian approach, the conditional probability $P(C|E)$ of the conclusion C given the evidence E is of interest. (We follow the notation of [49].) To use Bayes rule

$$P(C|E) = P(E|C)\,P(C) / P(E) \quad ,$$

the prior $P(C)$ and the conditional probability $P(E|C)$ must be available. However, the evidence E usually has a number of components $E_1, \ldots, E_n$. For a large numer of components, it is impractical to obtain the joint density. Consequently, one makes the simplifying assumption that the events $\{E_i|C\}$ are independent. The required information can then be computed from $P(C)$, $P(E_i|C)$, and $P(E_i|\tilde{C})$.

The Bayesian approach has been used in expert system shells such as KES [40] and a micro-based system discussed by Cox in [16]. Some criticism of this approach has pointed to the difficulty of assigning prior probabilities accurately and the weakness of the independence assumption.

Fuzzy logic is another approach advocated for expert systems [55],[38]. Its proponents claim that this methodology can serve as a faithful model of human thought processes dealing with uncertainty and yet still be based on a consistent theoretical

foundation. To see the role of fuzzy notions, consider the following rule: "If the percentage of MBA students in the course is high and the overall mathematical background is minimal, then the text should be case-oriented". While certainty factors weaken the statements indicating high percentage and minimal background by attaching numerical factors to them, fuzzy logic takes the predicates high and minimal to be fuzzy (without absolute numerical boundaries). In this way, uncertain inferences can be handled by the rules of fuzzy logic. A good description of fuzzy logic in the context of expert systems, together with an annotated bibliography of the relevant sources, is given by Negotia [38]. The use of fuzzy notions in the expert system shell REVEAL is described in tutorial fashion in chapter 9 of [16] and [39]. References [16] and [49] discuss other approaches to uncertain reasoning in addition to the preceding two.

The short discussion given here does not reflect some of the current research efforts in seeking more solid mathematical foundations for uncertain reasoning in expert systems. While such research will no doubt continue and bear fruit, some believe that this line of research is only of incidental importance. For instance, Hayes-Roth [22] notes that we are dealing with subjective assessments and that different methods of handling uncertainty work equally well if they can mimic human performance.

Alternative Designs for Expert Systems

Section 2 focused on a simple structure for an expert system, namely that of a production system with backward chaining. For introducing the expert system technology, we focused on this

particular structure in view of its prevalence in microcomputer-based shells. However, this is by no means the only structure that has been used. Indeed, the organization of the KB and the inference engine can both be designed in different ways. In what follows, we briefly list some of the possible variants.

Even in a rule-based system, there might be a need for special organizations of the data in the KB. For example, one would want to perform an easy check of the premise in a rule of the form "If 3 out of the 6 conditions of list L are satisfied, then send file to the auditor" (see [49]). In other cases, arrangements must be made to enable the program to read in and combine information from different measurement instruments. These constraints generally lead to an enrichment of the KB structure beyond rules and simple facts.

We have mentioned that inputting rules for a rule-based system is a cumbersome process. One could argue that, without the help of a knowledge engineer, it is hard for the user to formalize all the rules. This has generated some interest in **example-based** systems in which the user enters examples and the program constructs rules consistent with these examples. The examples may be entered in a decision matrix format (i.e., attribute-value tuples). Given n attributes, each entry corresponds to an (n+1)-tuple [a(1), a(2), ..., a(n), value]. Clearly, this approach simplifies the process of knowledge acquisition which, one could argue, is an especially important consideration in microcomputer-based tools.

The shell EXPERT-EASE uses the example-based approach and constructs a decision tree, which may be viewed as a collection of rules,

for arriving at a conclusion from a set of input values for the attributes. The same basic approach, enhanced by added capabilities, is used in TIMM-PC, another expert system shell. This system can generate example cases for the user to attach values to and, more importantly, can generalize from a given set of rules and also eliminate redundant ones.

These capabilities can be considered as first steps towards the goal of automating knowledge acquisition. Larger expert systems such as META-DENDRAL and AQ11 also have general inductive inference capabilities but there are still some barriers in the way of cost-effective use of such methods in commercial systems [23].

Other Knowledge Representation Schemes

Apart from production systems, other schemes may be used to represent knowledge. Better known examples include frames, semantic networks and formal logic (or predicate calculus). Table 1 summarizes some characteristics of these schemes. Of course, the various schemes should not be considered mutually exclusive. One can incorporate ideas from, say, semantic nets into a rule-based system or use a combination of different approaches. Translations from one scheme to another are also possible but may lead to running time inefficiencies. For example, Nau [37] cites a rewrite of HEARSAY-II as a production system that ran 100 times slower than the original version. We cannot embark on a detailed discussion of these in this paper but refer the reader to Volume 2 of [4], [19], [37], and part seven of [6].

The choice of a representation scheme is heavily influenced by the application domain. For example, semantic networks have

Table 1. Schemes for Representing Knowledge

REPRESENTATION SCHEME	FORM	ADVANTAGES	DISADVANTAGES	SYSTEMS BASED ON SCHEME
PRODUCTION SYSTEMS	Production Rules	- Highly modular - Easy to understand - Easy to add to or modify	- Control flow hard to follow - Hierarchies hard to capture - Inefficient for large systems	MYCIN XCON (R1) DRILLING ADVISOR
SEMANTIC NETWORKS	Nodes representing objects or descriptions joined by links	- Flexible - Easy to capture hierarchy - Easy to trace associations	- Exception handling is difficult - Meaning attached to nodes may be ambiguous	PROSPECTOR INTERNIST SCHOLAR
FRAMES	Set of slots and associated values represent an object	- Easy to include default information - Can detect missing values - Can see if frame matches available data	- Much of the work is still at experimental stage	PIP CENTAUR
FORMAL LOGIC	Statements of first-order logic (predicate calculus)	- Precision - Correctness of conclusions - Completeness	- Fails on large datasets - Combinatorial explosion - Separation of representation and processing	Theorem Proving Systems

been used to represent relations of causality in applications where this is a key notion (such as fault or disease diagnosis). In the diagnostic expert system designed by Reggia and his coworkers [41], there is description of each disease listing its manifestations (symptoms) and its setting (such as age and sex). It is easy to justify this representation scheme since it conforms to the physicians organization of diagnostic knowledge. Naturally, such an approach must be accompanied by an appropriate inference mechanism; the one used in [41] is based on a set covering approach which sequentially chooses the minimal set of diseases that can account for all of the observed manifestations.

To take another example, Davis has used a model of causal interaction in an expert system for electronic troubleshooting (see [7]). One reason he states for not using a rule-based system is that there would be very little carry-over of the rules from one device to another. By focusing on the causal pathways, however, one can observe greater commonality across devices.

The preceding example illustrates a general principle that induces researchers to go beyond production systems, namely, the distinction between deep and surface knowledge [20]. Surface knowledge captures certain associations without entering the deeper causal or structural connections between the facts. For instance, in medical diagnosis one can associate diseases with symptoms without considering the deeper notions of physiological pathways or the temporal evolution of ailments. Production systems only capture surface knowledge and thus may (unkindly) be likened to the student relying on rote learning rather than a profound understanding of the subject. It is generally agreed that more complex representation schemes are needed to capture deep knowledge, thereby

complicating the updating and modification of the system. This exemplifies one of the many tradeoffs involved in choosing an appropriate scheme.

Different Task Types

Our discussion so far has centered on expert systems whose main function is selection, diagnosis, or classification. In addition to these, one text [23] mentions the tasks of interpretation (of experimental data), monitoring (devices), forecasting or prediction, design (of complex equipment or layouts), and tutoring. Early expert systems developed in each of these areas are reviewed in [4] while Harmon and King [19] list some of the later applications. Gevanter [17] provides a table that groups existing expert systems by function.

While we cannot survey each of these task types, we must mention the area of intelligent tutoring and training systems that may be of use in operations research (see [15], [45], and part 8 of [6]). Along more speculative lines, Michie [34] mentions "knowledge refinery" as another function of expert systems. This involves using such systems to restructure the input domain knowledge into a more compact and understandable form.

4. IMPLEMENTATION AND EVALUATION OF EXPERT SYSTEMS

This section addresses the issues relating to the implementation of an expert system and the evaluation of its performance. Some literature on this subject is already available [23], [49], but most of the experience has been with larger expert systems developed on the mainframe. These systems are characterized by long development times and high costs, say of the order of 10-25 man-years and 1-2 million dollars [37], [24]. Consequently, their development has been observed and documented with some care. While a good part of these observations also applies to microcomputer-based systems, the smaller systems do present certain new issues and require a change of outlook. One of the few general discussions of implementing small systems may be found in [19].

The implementation process may be divided into the following phases:

1) selection of an appropriate problem;
2) selection of an appropriate development tool;
3) development of a prototype system;
4) development of the complete system;
5) evaluation and refinement of the expert system.

The first phase involves determining the suitability of expert systems for the application at hand. Here, past applications of expert systems provide a good guide. Characteristics of tasks suitable for expert system modeling include the following:

* presence of symbolic and subjective factors affecting the decision-making;
* a relatively narrow domain for the task;
* the dependence of expert performance on specific knowledge (as opposed to common sense or general knowledge);

* the appropriate level of complexity of the task (it should not be too easy or too difficult for the human).

The second phase involves the choice of a building tool. For microcomputer-based applications, this would most likely entail selecting a commercially available shell. Apart from the different functional features, available shells differ in the amount of computer expertise they require from the user.

The third phase of building a prototype has been crucial to the development of some large expert systems. Even for a smaller system, the prototype helps the user to decide on the structure of the knowledge base before spending a large amount of time entering data. Previous experience has shown that developing the prototype accelerates the process of knowledge acquisition. Experts find it easier to criticize an existing program or provide exceptions to its rules than to formalize rules in the abstract [49]. Moreover, a prototype system can help sustain the expert's interest. Upon completion, the prototype can be expanded to the full system.

Before commenting on the evaluation of the complete system, we should remark that there is a close link between the evaluation and refinement tasks in the course of developing an expert system. Evaluation reveals cases not handled by the system's rules and causes new rules to be added. Next, the interaction of the new rules with the existing ones has to be evaluated since the new rules may have unexpected negative effects on parts of the system. Some researchers including [49] and [11] suggest using a representative database of test problems that can be used throughout

the development process. After each modification, the system would be re-run on this testbed to assess the effect of the changes.

Evaluation of expert systems

During the development of MYCIN, a sizeable research effort focused on the issue of evaluating an expert system. The observations of well-known authorities on this question are gathered in Chapter 8 of [23] and part 10 of [6]. Comments on specific systems may be found in [3] and [5]. We should remark that the wealth of evaluation research on MYCIN is probably unmatched by other systems. In commercial implementations in particular, the issues of proprietary rights have limited the dissemination of the nature and results of evaluation studies.

As mentioned in [49], the evaluation question has been approached in two ways-- the anecdotal approach and the empirical one. The former is a collection of experiences with particular systems, recounted with the aim of drawing lessons about the elements of success and failure in each case (this is similar to the case-study method). The latter approach involves setting up experimental designs for testing the system over sample cases. Most evaluations of expert system fall into the first category. The following issues should indicate the difficulties that stand in the way of evaluation studies.

1) What characteristics should be evaluated? The performance of the system has been the main characteristic of interest. However, the system's discourse or ease-of-use may also be key to its acceptance.

2) How should performance be evaluated? Due to the nature of expert system applications, it is sometimes hard to define a "gold standard" against which to compare the system's performance. For example, a match between the conclusions of the system and the expert may be hard to obtain. Indeed, different experts may disagree on certain details or both the program and the expert may be wrong. In evaluating performance, should one look only at the conclusion or should the program's line of reasoning be evaluated as well? What form should the evaluation take when the system provides multiple (as opposed to unique) answers?

3) How should the test problems be selected? The fact that the realism of real-world exceptions and irrelevencies can seriously affect the performance of an expert system is well-known. However, in certain areas, the supply of realistic studies may be very limited. In the case of PROSPECTOR, for instance, there are only a small number of known ore deposits to draw upon and the time between initial and final characterizations of the deposit could be long [11]. Similar problems occur with rare diseases and other cases when sampling costs are high.

4) How should one evaluate the program's mistakes? In judgmental areas, it is interesting to observe the type of mistakes an expert sytem may make. One is reminded that the work of Piaget on developmental psychology was prompted by the patterns of mistakes (not the correct responses) in IQ tests by children. The same search for error patterns occurs in intelligent tutoring systems but the implications for evaluation studies appear to be unexplored. Clearly, this issue also relates to the requirement that expert systems "degrade gracefully" [8].

In any case, the difficulty and pitfalls associated with evaluation studies is well-recognized [23]. One could ask what influence a proliferation of microcomputer-based systems may have on evaluation questions. Initially, at least, the enthusiasm for building small systems will no doubt overshadow validation studies. The user will have a higher degree of involvement with the system from the start and may in fact help in its design. As the task modeled may be relatively simple in a small expert system, the evaluation of its performance may become a simpler matter. Instead, the user interface will probably play a larger role in the evaluative process and its design will assume a greater importance. Finally, some methods of evaluating the large number of commercial expert system shells must unfold to help the user select the appropriate tool for his application.

Languages

We conclude this section with some brief remarks on languages used in expert system tools. Traditionally, the favorite language for developing AI software has been LISP [52], [53]. The pattern-matching capability of this language has been a key element in the development of early expert systems.

Another language suitable for symbolic processing is PROLOG. The control structure of PROLOG is logical inference, meaning that it is oriented towards deriving conclusions from given facts. (A short tutorial on this feature of the language may be found in Chapter 7 of [16].) The recent interest in this language has been fueled in part by the reports of its prospective use in Fifth Generation machines [14]. However, Forsyth [16] lists a number of disadvantages of PROLOG including the limited power of the programmer over the control (or search) strategy. Other remarks on LISP and PROLOG can be found in [46].

One source states that "high-level languages like PASCAL, ADA, and C are acceptable for the delivery system, but for prototyping, a language like LISP or PROLOG is preferred" [33]. This comment reflects the increasingly common strategy of developing the system with one language and translating the completed system into another. Thus CATS-1 was translated from LISP into FORTH so it could run more efficiently and PUFF, initially coded in LISP, was translated into BASIC. Among microcomputer-based shells, EXPERT 2 uses FORTH and TIMM-PC uses FORTRAN.

In suummary, there has been an increasing trend to use more common languages such as PASCAL or FORTRAN for expert systems, particularly in their microcomputer implementations. Simultaneously, LISP is becoming more readily accessible and popular [52]. Feigenbaum has pointed out the feasibility of having highly efficient list processing in a personal computer environment [13]. While the cost of such a machine is currently too high, many predict that efficient logic or list processing may soon be commercially available for microcomputers at a reasonable cost. It would be interesting to witness the impact of this hardware for microcomputers on the future of expert systems.

5. EXPERT SYSTEMS AND OPERATIONS RESEARCH

Are expert systems and operations research (OR) going to remain two separate areas of research or will interesting interfaces between the two areas emerge? In the past, certain hopes for integrating AI and operations research have failed to materialize. Expert systems, however, are viewed as "applicable AI" and their potential role in finance, accounting, and production is currently the focus of much interest leading to a number of ongoing research efforts.

Reading certain accounts of expert systems can easily lead to the impression that there is a natural distance between expert systems and the algorithmic tools of operations research. The following two quotations exemplify this notion.

> "Unlike a systems analyst, who formulates an algorithm to solve a client's problem, the knowledge engineer seeks to capture the existing problem-solving method" [22].

> "Not only should the the system obtain the performance of a human expert but it should also explain results in a manner similar to the human expert... this would exclude most purely statistical models, mathematical simulations, and numerical approximation models" [25].

Nevertheless, these remarks do not preclude hybrid systems where an expert system interacts with a more traditional mathematical or statistical package. Indeed, after citing certain expert systems that use large mathematical packages as subroutines, Kastner and Hong state that "one of the most obvious first applications of expert systems to OR would be in this spirit" [25].

In his interesting article, Hahn [18] maps the potential connections between statistical methods and expert systems. A good part of the observations of this paper also apply to the interface with operations research. In what follows, we shall build on his paper to suggest some possible directions for using expert systems within operations research.

1) Choice of Models. An expert system can help the user to define appropriate models for his problem. Through a series of questions, the main problem characteristics affecting the choice of a model can be extracted from the user. For example, in an area such as location theory where there are many known results which apply under various assumptions, the program could point to an appropriate subset and then attempt to refine the models by further questioning.

2) Model Generation. The notion of interactive model generation is not new to operations research. With expert systems, it may be possible to introduce some added intelligence into the process. Queries could prompt the user to add side constraints to an optimization model or help the user define the components of a complicated simulation model.

3) Search for Solutions. In certain applications of OR, the solutions generated must satisfy a variety of constraints that are not easily quantified, or would overly complicate the problem if included. Moreover, the trade-off between "hard" and "soft" constraints involves subjective evaluations. The cable TV network design problem described in the paper by Frisch in this issue is one example of a complicated problem with many objectives. Vehicle routing in the presence of all "real-life" constraints

is another. The solution produced by the model generally violates some of these constraints to varying degrees. Conceivably, expert systems could elicit the various subjective criteria from the user and interact with the model (maybe through parameter settings) to drive towards their satisfaction.

4) Interpretation of Data. Feigenbaum [13] describes an example where the incorporation of a knowledge-based system into a standard statistical tool for signal processing cut the costs by two orders of magnitude. The lesson is that intelligent systems could significantly lower the costs of data interpretation since, in some sense, one has a better idea of what one is looking for.

5) Open Simulations. STEAMER [50] is an intriguing example of how a simulation tool can be endowed with an intelligent superstructure to help familiarize the user with a complex system. By changing the various inputs and parameters of the system through the expert system, the user can develop a mental model of how the complex system operates. This approach may be viewed as an extended user-friendly capability for performing sensitivity analysis.

6) Training Systems. We have already mentioned the potential of intelligent training systems [15],[45] for both instruction and model- building in OR. Coupled with a system such as STEAMER, instructional systems may be useful to train the users of complicated models, such as energy and equilibrium models.

7) Introspective Models. One goal of current research in AI is to build systems that can reason about their own reasoning

powers and modifdy it if necessary. A simple example of this is given by the SEEK system as described in [49]. This program can suggest experiments to evaluate the performance of certain rules within its knowledge base. In our view, the way SEEK accomplishes this offers interesting possibilities in OR where many algorithms are fine-tuned by performing extensive empirical tests of various options or parameter settings. Conceivably, an expert system could help in this process. The system would be especially valuable if it can detect patterns in the outputs of the various experiments, which humans might not easily see.

These are only some of the possible directions for the use of expert systems in operations research. It has been remarked that the "consultant" paradigm may not be the most appropriate one for expert systems used in statistical methods, and that "research assistant" is a better model [20]. The implication that expert systems would serve as aids to the understanding, formulation, and interpretation of the mathematical models may be equally applicable to operations research. Clearly, for a successful match, both expert system developers and operation researchers have to modify their existing notions of what the role of a model is.

6. MICROCOMPUTER-BASED EXPERT SYSTEM SHELLS

Most of the expert systems discussed earlier in this paper were developed using variants of the LISP programming language on mainframes or minicomputers. Lehner and Barth [27], however, focus on microcomputer-based expert systems such as AL/X (developed for British Petroleum), DELTA (developed by General Electric Corp.), and GEN-X (also developed by General Electric Corp.). They discuss several implementations in detail and then address the important question: "Why use microcomputers for expert systems implementation?"

There are a number of answers to the above question. First, microcomputers are inexpensive, widely available, and general-purpose machines. As a result, individuals or companies interested in experimenting with expert systems don't have to purchase a Symbolics 3600, a TI Explorer, or a comparable LISP workstation which can easily cost between $50,000 and $75,000. In addition, the dedication of a microcomputer to an expert system project is feasible whereas dedicating a more costly machine would raise major objections. Since microcomputers are so inexpensive, they can be distributed over a wide geographic region, e.g., one to each district or county. Where expertise is rare, this might be especially attractive and cost-effective as well. A second key issue is transportability. Both hardware and software can be moved easily from one location to another depending on the demand. Finally, there is the widely-recognized interactive convenience of the microcomputer. Since a session with an expert system ("consultant") should be an interactive experience, the match is a natural one (computation speed and storage aside).

With the above answers in mind, a number of entrepreneurial AI companies have recently begun marketing microcomputer-based expert system shells. As mentioned in Section 1, these shells are tools that enable users to build expert systems. In our view, these shells should be of special interest to operations researchers who are well-trained in both model-building and implementing real-world systems. In Table 2, we provide current information on 15 such shells. For the most part, this table is self-explanatory. We do point out that prices range from under $100 to as much as $10,000. Some systems are purely symbolic, whereas others allow numeric computations within rules; these computations can be extremely important in certain types of business applications (e.g., interpretation of time series). Table 2 is based, primarily, upon a

Table 2. Microcomputer-Based Expert System Shells*

PACKAGE	ADVISOR	ES/P ADVISOR	EXPERT-EASE
Distributor	UME Inc. 275 Magnolia Ave. Larkspur, CA 94939	Expert Systems International 1150 First Ave. King of Prussia, PA 19406	Expert Systems, Inc. 868 West End Avenue Suite 3A New York, NY 10025
Contact	Bill Moulton (415) 924-3644	Angelos Kolokouris (215) 337-2300	Jeffrey Milman (212) 662-7206
Price	$100.	$895.	$595.
Demonstration Diskette	No	$65.	$65.
Knowledge Representation	Rule-based	Rule-based	Example-based
Inference Technique	Forward and backward chaining	Backward chaining	Seeks to build most compact decision tree
Certainty Factors	No	No	No
Hardware Requirements	Apple II Commodore 64 Atari 800	IBM PC with 256KB DOS 2.0	IBM PC with 128KB DOS 2.0
Numeric Computation within Rules	No	No	No
Language	ASSEMBLY	PROLOG-1	PASCAL
Miscellaneous Comments	IBM PC version forthcoming	Open-ended architecture; Easy for experienced programmers to build upon	Uses spreadsheet format

*Accurate as of 7/15/85.

Table 2. Microcomputer-Based Expert System Shells, (continued)

PACKAGE	EXPERT-EASE†	EXPERT EDGE	EXPERT 2
Distributor	Human Edge Software 2445 Faber Place Palo Alto, CA 94303	Human Edge Software 2445 Faber Place Palo Alto, CA 94303	Miller Microcomputer Se 61 Lake Shore Rd. Natick, MA 01760
Contact	Lisa Yauger (800) 624-5227	Lisa Yauger (800) 624-5227	A. Richard Miller (617) 653-6136
Price	$695.	$795.	$250.
Demonstration Diskette	No	No	No
Knowledge Representation	Example-based	Rule-based	Rule-based
Inference Technique	Seeks to build a most compact decision tree	Forward and backward chaining	Backward chaining
Certainty Factors	No	Yes	No
Hardware Requirements	IBM PC with 128KB DOS 2.0	IBM PC with 256KB DOS 2.0	TRS-80 Model I, III, 4 & IBM PC, at least 64KB
Numeric Computation within Rules	No	Yes	Yes, through callable subroutines
Language	PASCAL	C	MMSFORTH
Miscellaneous Comments	Uses spreadsheet format	Available starting 8/1/85	Unfriendly package

† Also available from Jeffrey Perrone & Associates, 3685 17th St., San Francisco, CA 94114, (415) 431-9562.

Table 2, Microcomputer-Based Expert System Shells, (continued)

PACKAGE	EXSYS†	INSIGHT†	KDS
Distributor	EXSYS, Inc. P.O. Box 75158 Contract Sta. 14 Albuquerque, NM 87194	Level 5 Research 4980 South A-1-A Melbourne Beach, FL 32951	KDS Corporation 934 Hunter Road Wilmette, IL 60091
Contact	Dustin Huntington (505) 836-6676	Henry Seiler (305) 729-9046	Barbara or Bill Wallace (312) 251-2621
Price	$295.	$95.	$945.
Demonstration Diskette	$10.	No	$150.
Knowledge Representation	Rule-based	Rule-based	Example-based
Inference Technique	Backward chaining	Forward and backward chaining	Forward and backward chaining
Certainty Factors	Yes, three scales	Yes	Yes
Hardware Requirements	IBM PC with 256KB DOS 2.0	IBM PC with 128KB DOS 2.0	IBM PC with 512KB recommended for development DOS 2.0
Numeric Computation within Rules	Yes	No	No
Language	C	PASCAL	ASSEMBLY
Miscellaneous Comments	Remarks and references can be attached to rules	INSIGHT 2 is now available for $485.--Interfaces with DBASE2	Package was previously called PATHFINDER

†Also available from Jeffrey Perrone & Associates, 3685 17th St., San Francisco, CA 94114, (415) 431-9562.

Table 2. Microcomputer-Based Expert System Shells, (continued)

PACKAGE	KES	M.1
Distributor	Software Architecture & Engineering, Inc. 1500 Wilson Blvd. Suite 800 Arlington, VA	Teknowledge, Inc. 525 University Ave., #200 Palo Alto, CA 94301
Contact	Simon Blackwell (703) 276-7910	Judy Harris (415) 327-6600
Price	$2,000.	$10,000. for commercial use $ 1,250. for university research [M.1a, a subset of M.1, is availat for commercial use for $2,000.]
Demonstration Diskette	See Building Your First Expert System with Micro-PS by Nagy, Galt, and Nagy, published by Ashton-Tate ($30.)	Yes, it's free
Knowledge Representation	Three Types: Rule-based, Frame-like, Statistical pattern classification scheme	Rule-based
Inference Technique	Three Options: Backward chaining, Minimal set cover approach, Bayesian statistical approach	Backward chaining with some forwa chaining
Certainty Factors	Yes	Yes
Hardware Requirements	IBM XT with 640KB, and 8087 coprocessor DOS 2.0	IBM PC with 128KB DOS 2.0
Numeric Computation within Rules	Yes	Yes
Language	LISP	PROLOG-1
Miscellaneous Comments	Has multiple knowledge bases and multiple inference engines	Rules can be generalized by the u of variables; M.1 justifies reco mendations effectively

Table 2. Microcomputer-Based Expert System Shells, (continued)

PACKAGE	MICROEXPERT	PERSONAL CONSULTANT	REVEAL
Distributor	MicroExpert Systems R.D. 2, Box 430 Nassau, NY 12123	Texas Instruments, Inc. P.O. Box 225012 Mail Station 57 Dallas, Texas 75265	McDonnell Douglas Advanced Products Division 20705 Valley Green Drive Cupertino, CA 95014
Contact	Beverly Thompson (518) 766-3982	Harlow Russell (512) 250-7858	Walter Eissmann (408) 446-6236
Price	$50.	$3,000.	$4,500.
Demonstration Diskette	No	Yes, it's free	No
Knowledge Representation	Rule-based	Rule-based	Rule-based
Inference Technique	Backward chaining	Backward chaining with some forward chaining	Fuzzy sets technology
Certainty Factors	No	Yes	Yes
Hardware Requirements	IBM PC with 128KB DOS 2.0 or Apple II with 64KB	TI Professional with 512KB IBM PC with 512KB DOS 2.0	IBM XT with 640KB, and 8087 coprocessor DOS 2.0
Numeric Computation within Rules	Yes	Yes	Yes
Language	PASCAL	IQLISP	FORTRAN
Miscellaneous Comments	Package marketed by McGraw-Hill (800) 628-0004	Uses the same rule structure as the EMYCIN system developed at Stanford Univ.	Seeks to combine capabilities of Expert Systems with those of Decision Support System

Table 2. Microcomputer-Based Expert System Shells, (continued)

PACKAGE	TIMM-PC	TOPSI
Distributor	General Research Corp. 7655 Old Spring House Rd. McLean, VA 22102	Dynamic Master Systems, Inc. P.O. Box 566456 Atlanta, GA 30356
Contact	Nancy Nay (703) 893-5900	David Smith (404) 565-0771
Price	$9,500.	$ 75. for prototyping version $175. for production version
Demonstration Diskette	No	No
Knowledge Representation	Example-based	Rule-based
Inference Technique	Nearest neighbor algorithm	Forward chaining
Certainty Factors	Yes	No
Hardware Requirements	IBM XT or AT with 640KB DOS 2.0	IBM PC with 256KB DOS 2.0
Numeric Computation within Rules	No	No
Language	FORTRAN	PASCAL
Miscellaneous Comments	Dial-in mainframe demonstration is available at no charge	First microcomputer implementatio of OPS5, originally developed on VAX at Carnegie-Mellon Universit

search of the popular literature, a reading of the package brochures, and a telephone survey that we conducted. In addition, we have worked with 6 of the 15 packages.

The careful reader will observe with some surprise that there are three distributors of EXPERT-EASE. We were unable to determine precisely how this has come to pass. Each distributor tells a different story.

As with the compendium presented in Chapter 8 of Harmon and King [19], the purpose of Table 2 is to be as informative as possible. Comparing shells is subjective and dangerous for the following obvious reasons:

1. One needs to work closely with all 15 shells before comparisons can be valid.
2. Some shells are too expensive to easily purchase;
3. Different shells do different things and tackle slightly different selection problems;
4. Documentation, vendor support, and user-friendliness are so important and yet so difficult to measure;
5. Comparing a $100 package with a $10,000 package raises a host of complicated tradeoff questions;
6. The shells are continuously being updated.

Despite these concerns, there are several factors to consider when deciding on the purchase of a shell. Of course, cost is a primary issue and the shell must be compatible with available microcomputers. We remark that, based upon our research, cost is not always the best indicator of value. The size of the knowledge base is another major issue. Microcomputer shells can currently handle up to several hundred rules comfortably. Microcomputer speed and available internal memory become bottlenecks beyond this point. The knowledge representation scheme and programming language are important factors as well. The former should typically be compatible with the existing knowledge base, and the latter should be widely used. Experience with the programming language is a definite advantage. Help and explanation facilities along with trace and justify capabilites help make an expert system easy to use. So too does good documentation. Finally, we point out that if the existing knowledge base contains "probabilistic" or "fuzzy" rules then the shell should support this feature; if rules are likely to have numeric as well as symbolic components, then the shell should allow this also.

As mentioned earlier in this section, we did manage to work with 6 of the 15 shells--EXPERT-EASE, EXPERT 2, EXSYS, INSIGHT, M.1, and TIMM-PC. This subset is fairly representative of the entire set of shells with regard to most features. Based upon several sessions with each of the six, we compiled a list of strengths and weaknesses (see Table 3).

From Table 3, we are able to offer some tentative suggestions. For operations researchers who want to learn about or experiment with some small expert systems, EXSYS might be a very appropriate product. It has most of the features that the more costly shells have and it is not too difficult to learn. In addition, we might suggest the purchase of the demonstration diskette for EXPERT-EASE. This provides a feel for what an example-based expert system looks like and how it works.

Finally, for those with university affiliations, we might recommend M.1 (see Table 2) as the most powerful microcomputer-based expert system shell that we have seen. Commercial users will probably want to perform a first-cut cost-benefit analysis before spending full price ($9,500 to $10,000) for TIMM-PC or M.1. We expect these prices to drop significantly in the coming year. We also would not be surprised to see PERSONAL CONSULTANT and REVEAL selling for under $2,000 each in the near future. Furthermore, potential developers should keep in mind that new shells are emerging in the microcomputer marketplace very rapidly and some of these are expected to have excellent capabilities.

Additional sources of information on shells or on implementing expert systems on microcomputers are available (e.g., see [24], [26], [28], [31]).

Table 3. Evaluation of Microcomputer-Based Expert System Shells Tested

PACKAGE	STRENGTHS	WEAKNESSES
EXPERT-EASE	1. moderately priced 2. extremely easy to use 3. excellent users manual	1. cannot provide explanations 2. doesn't incorporate uncertainty 3. attribute values must be either logical or integer 4. doesn't handle ranges (e.g., $5 \leq x \leq 10$)
EXPERT 2	1. inexpensive	1. very restrictive licensing agreement 2. poor documentation 3. does not greatly facilitate the building of an expert system
EXSYS	1. inexpensive 2. good vendor support 3. handles uncertainty 4. "Why" and "How" questions are supported 5. numeric computation within rules is supported	1. documentation is not bad, but could be improved
INSIGHT	1. very inexpensive 2. good documentation 3. handles uncertainty 4. line of reasoning can be traced, if desired	1. numeric computation within rules is not allowed 2. no internal text editor--must use outside word processor to create rule base and then compile
M.1	1. documentation is comprehensive 2. excellent demonstration expert systems are provided 3. handles uncertainty 4. justifies recommendations effectively and traces line of reasoning 5. supports "variable" rules which essentially allow the user to express many rules as one	1. expensive 2. numeric calculations are somewhat limited 3. knowledge base must be created by outside word processor
TIMM-PC	1. compression and generalization of the knowledge base is supported—very impressive 2. even when pieces of information are missing, package will provide a recommendation 3. handles uncertainty 4. prompt-driven	1. numeric computation within rules is not allowed 2. expensive 3. number of allowable outcome choices is small

7. TRENDS AND FUTURE DIRECTIONS

Expert systems have certainly attracted a great deal of attention in the last couple of years. We have tried in this paper to indicate some of what is behind the excitement, especially as it relates to microcomputers. In this final section, we look ahead in time, say, towards 1990. Rather than offer predictions, however, we raise a number of questions which we feel to be relevant:

1. Will we see the widespread commercial use of small, specialized expert systems as envisionsed by Harmon and King [19]?
2. Alternatively, in five years, will the view be that expert systems were oversold, prematurely (see Martins [30]).
3. Will expert systems prove to be successful in training applications?
4. Current state-of-the-art research on expert systems involves the inclusion of "deep knowledge" along with "learning" capabilities into expert systems. Will this work filter down to the microcomputer-based shells?
5. Will expert systems and robotics come together, as suggested by Michie and Johnston [36]?
6. Due to their memory-intensive nature, will expert systems drive up the memory requirements of the microcomputer (see [33])?
7. With the proliferation of microcomputer shells, will the need for LISP machines and knowledge engineers diminish?
8. Will hybrid systems, combining optimization modules and specialized expert systems (e.g., in vehicle routing), find acceptance (see [25])?
9. With so many expert systems tools available, will operations researchers play a leading role in implementing commercially successful systems?

These questions are fun to ask, but of course only time will tell. In the interim, expert system shells are available in all shapes and sizes, and we strongly urge interested readers to take advantage of this opportunity to experiment.

Acknowledgments

The authors would like to thank the Westinghouse Defense and Electronic Systems Center (Baltimore) for its generous support. In addition, we are grateful to Bill Dorris, Mark Jones, and Snehamay Bandyopadhyay for their helpful comments.

REFERENCES

[1] T. Alexander, Why computers can't outthink the experts. Fortune 110 (4), 110-116 (1984).

[2] S. J. Andriole (Ed.), Applications in Artificial Intelligence. Petrocelli Books, Inc. , Princeton, NJ (1985).

[3] J. Bachant and J. McDermott, R1 revisited: Four years in the trenches. AI Magazine 5 (3), 21-35 (1984).

[4] A. Barr and E. A. Feigenbaum, The Handbook of Artificial Intelligence. Volumes 1-3, William Kaufmann, Los Altos, California (1982).

[5] A. Basden, On the application of expert systems. pp. 59-75 of [7].

[6] B. G. Buchanan and E. H. Shortliffe, Rule-based Expert Systems. Addison-Wesley, Reading, MA. (1984).

[7] M. J. Coombs (Ed.), Developments in Expert Systems. Academic Press, London (1984).

[8] R. Davis, Expert systems, where are we? And where do we go from here? AI Magazine 3 , 3-22 (1982).

[9] R. Davis, Amplifying expertise with expert systems, pp. 17-40 in [54].

[10] R. O. Duda, J. Gaschnig and P. E. Hart, Model design in the PROSPECTOR consultant system for mineral exploration. pp. 153-167 of [35].

[11] R. O. Duda and E. H. Shortliffe, Expert systems research. Science 220 (4594), 261-268 (1983).

[12] E. A. Feigenbaum, Themes and case studies of knowledge engineering. pp. 3-25 in [35].

[13] E. A. Feigenbaum, Knowledge engineering: the applied side. pp. 37-55 in [21] .

[14] E. A. Feigenbaum and P. McCorduck, The Fifth Generation. Addison-Wesley, Reading, MA (1983).

[15] J. D. Fletcher, Intelligent instructional systems in training. pp. 427-451 in [2].

[16] R. Forsyth (Ed.), Expert Systems: Principles and Case Studies. Chapman and Hall, London (1984).

[17] W. B. Gevanter, Expert systems: limited but powerful. IEEE Spectrum, 20 (8), 39-45 (1983).

[18] G. J. Hahn, More intelligent statistical software and statistical expert systems: future directions. The American Statistician 39 (1), 1-16 (1985).

[19] P. Harmon and D. King, Expert systems. Artificial Intelligence in Business. Wiley, New York (1985).

[20] P. E. Hart, Directions for AI in the eighties. SIGART Newsletter, No. 79, 11-16 (1982).

[21] J. E. Hayes and D. Michie (Eds.), Intelligent Systems: The Unprecedented Opportunity. Halsted Press, Wiley, New York (1983).

[22] F. Hayes-Roth, The knowledge-based expert system: A tutorial. Computer 17 (9), 11-28 (1984).

[23] F. Hayes-Roth, D. A. Waterman and D. B. Lenat (Eds.), Building Expert Systems. Addison-Wesley, Reading, MA (1983).

[24] E. Horwitt, Exploring expert systems. Business Computer Systems 4 (3), 48-57 (1985).

[25] J. K. Kastner and S. J. Hong, A review of expert systems. European Journal of Operational Research 18, 285-292 (1984).

[26] P. Kinnucan, Software tools speed expert systems development. High Technology 5 (3), 16-21 (1985).

[27] P. E. Lehner and S. W. Barth, Expert systems on microcomputers. pp. 109-123 of [2].

[28] B. Lemly et al. , Artificial expertise: Intelligent software for problem solving. PC Magazine 4 (8), 108-166 (1985).

[29] D. B. Lenat, Computer software for intelligent systems. Scientific American 251 (3), 204-213 (1984).

[30] G. R. Martins, The overselling of expert systems. Datamation 30 (18), 76-80 (1984).

[31] P. McWhite and G. Paul, TIMM: A Fortran-based expert systems applications generator, General Research Corp. , McLean, VA (1985).

[32] W. F. Messier, Jr. and J. V. Hansen, Expert systems in accounting and auditing: A framework and a review. in Decision Making and Accounting : Current Research. E. Joyce and S. Moriarty (Eds.), University of Oklahoma Press (1984).

[33] R. H. Michaelsen, D. Michie and A. Boulanger, The technology of expert systems. BYTE 10 (4), 303-312 (1985).

[34] D. Michie, A prototype knowledge refinery. pp. 57-69 in [21].

[35] D. Michie (Ed.) Expert Systems in the Microelectronic Age. Edinburgh University, Edinburgh (1979).

[36] D. Michie and R. Johnston, The Knowledge Machine. William Morrow & Co. , New York (1985).

[37] D. S. Nau, Expert computer systems. Computer 16 (2), 63-85 (1983).

[38] C. V. Negotia, Expert Systems and Fuzzy Systems. Benjamin/Cummings, Menlo Park, CA (1985).

[39] A. Okuma, Software and fuzzy logic let any good programmer design an expert system. Electronic Design 33 (8), 173-184 (April 4, 1985).

[40] J. A. Reggia, Knowledge-based decision support systems: Development through KMS. Report TR-1121, Department of Computer Science, University of Maryland, College Park (1981).

[41] J. A. Reggia, D. S. Nau and P. Y. Wang, Diagnostic expert systems based on a set covering model. pp. 35-58 in [7].

[42] E. Rich, Artificial Intelligence. McGraw-Hill, New York (1983).

[43] E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN. Elsevier, New York (1976).

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

[45] D. Sleeman and J. S. Brown (Eds.), Intelligent Tutoring Systems. Academic Press, New York (1982).

[46] E. Tello, The languages of AI research. PC Magazine 4 (8), 173-189 (1985).

[47] B. A. Thompson and W. A. Thompson, Inside an expert system. BYTE 10 (4), 315-330 (1985).

[48] R. Vedder and C. H. Nestman, Understanding expert systems: Companion to DSS and MIS. Industrial Management 27 (2), 1-8 (1985).

[49] S. M. Weiss and C. Kulikowski, A Practical Guide to Designing Expert Systems. Rowman & Allanheld, Towota, NJ (1984).

[50] M. D. Williams, J. Hollan and A. L. Stevens, An overview of STEAMER. Behavior Research Methods and Instrumentation 2 (13) (1981).

[51] P. H. Winston, Artificial Intelligence. 2nd edition, Addison-Wesley, Reading, MA. (1984).

[52] P. H. Winston, The LISP revolution. BYTE 10 (4), 303-312 (1985).

[53] P. H. Winston and B. K. P. Horn, LISP. 2nd edition, Addison-Wesley, Reading , MA (1984).

[54] P. H. Winston and K. H. Prendergast (Eds.), The AI Business. MIT Press, Cambridge, MA (1984).

[55] L. A. Zadeh, The management of uncertainty in expert systems, presented at the First Symposium on the Theory and Application of Expert Systems in Emergency Management Operations, FEMA/NBS, Washington, D.C. (April 1985).