1987

NASA/ASEE SUMMER FACULTY RESEARCH FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

EXPERT SYSTEM TECHNOLOGY

Prepared by:	Mary Ellen Prince, M.S.
Academic Rank:	Instructor
University and Department:	University of Alabama in Huntsville Department of Computer Science
NASA/MSFC:	
Laboratory:	Space Sciences
Division:	Astrophysics
Branch:	X-ray Astronomy
MSFC Colleague:	Martin C. Weisskopf
Date:	August 21, 1987
Contract No.:	The University of Alabama in Huntsville NGT 01-008-021

OBJECTIVES

The objectives of my research this summer were

1. To present a series of informal talks covering general topics in the area of artificial intelligence.

2. To concentrate in particular on the subject of expert systems, with an emphasis on their possible usefulness and practicality in scientific applications

This report will summarize some of the findings on expert systems.

INTRODUCTION

The emergence in recent years of expert systems as a new kind of computational tool is a trend that deserves close examination by every organization that uses computers for problem solving. To evaluate the utility of expert systems prospective users must understand what these systems are, how they are structured, and when they can produce beneficial results.

Briefly stated, an expert system is a computer program that attempts to reproduce the problem-solving behavior of an expert. Experts do not generally rely on domain axioms and principles when solving a problem. Instead, they tend to employ ready-made or empirical "know-how", compiled from past experience. They are able to view problems from a broad perspective and arrive at conclusions rapidly, using intuition, shortcuts, and analogies to previous situations. This experiential knowledge is called heuristic knowledge, and is what differentiates the expert from the merely competent.

Expert systems are a departure from the usual artificial intelligence approach to problem solving. Researchers have traditionally tried to develop general models of human intelligence that could be applied to many different situations. Expert systems, on the other hand, tend to rely on large quantities of domain specific knowledge, much of it heuristic. The reasoning component of the system is relatively simple and straightforward. For this reason, expert systems are often called knowledge based systems.

The body of this paper will attempt to expand on the ideas outlined above. Section 1 will explain the architecture of a typical expert system. Section 2 discusses the characteristics that make a problem a suitable candidate for expert system solution. Section 3 surveys current technology, describing some of the software aids that are available for expert system development. Section 4 discusses the limitations of this technology. The concluding section will attempt to predict future trends.

1. EXPERT SYSTEM COMPONENTS

Expert systems have three major components: a working memory, which contains facts about the current problem; the knowledge base, an integrated body of domain facts, heuristics, and relationships; and the inference engine, which is the reasoning and control mechanism. This organization is derived from earlier studies of production system models, which were originally proposed a general computational mechanism. [Davis and King, 84] Production systems consist of a data base, a set of rules, and a rule interpreter. They have been used for a variety of purposes, including compiler construction, formal language theory, and psychological modeling of human cognitive processes. Early expert system designers saw production systems as an elegant formalism for representing the mental activity of expert problem solvers.

Working Memory

Working memory corresponds in some ways to the data storage area of conventional programs. It is empty until a problem is proposed. At that time, the initial problem description and data are entered, perhaps by means of a consultation between the program and the user. As the program works toward a solution, additional facts are inferred or provided by the user. These are stored in working memory as well, as are any intermediate results concluded by the reasoning process.

Knowledge Base

The knowledge base is arguably the most important part of an expert system. Its structure is more complex that that of the working memory and its development is a collaborative effort between the domain expert and a knowledge engineer. The knowledge engineer is a trained professional whose responsibility is to help the expert express his know-how verbally. This is a difficult task, since one characteristic of expertise is the ability to solve problems without following a conscious step-by-step procedure. Once the knowledge is verbalized it must be encoded in some form and run on sample problems. If the solutions do not agree with the expert's solutions to the same problems, the knowledge must be modified. Thus, development of the knowledge base is an iterative procedure.

How the knowledge is represented is an important consideration. Heuristics are most commonly described by a set of production rules which have the format IF (condition) THEN (action). Rules are a convenient method of encoding "chunks" of knowledge, and are thought by many cognitive scientists to reflect the way the human brain organizes information. Rules are "triggered" when their conditions are satisfied by facts from the working memory. Once triggered, a rule may be "fired" by the inference engine. When a rule fires, its action causes a change in working memory, possibly triggering other rules in the process.

Domain facts can be represented in a number of different ways. Object-attribute-value (O-A-V) triples were used in many of the early systems. An object can be either a physical entity or a concept. Attributes are the general characteristics that describe an object, and values identify a specific instance of the object. For example the object "ball" might have attributes "size", "shape", and "color". A specific ball could then have descriptive attribute values such as "size" = "large".

Frames or units provide another way to describe objects. A single frame represents an object and contains a number of "slots". Slots are similar to, but more general than, the attributes in O-A-V triples. Two special slot types, superclass and subclass, provide a method of structuring factual information into a hierarchy. The object "ball" might be a member of the super-class "toys" and have as subclasses "baseball", "football", "golfball", etc. In the absence of knowledge about a particular object's attributes, values may be inherited, or passed down by default, from the frame which defines that object's superclass.

Logic programming languages such as PROLOG are another approach to knowledge representaion. Rules are expressed as logical propositions and facts as assertions. Built-in language features are then able to reason with this knowledge.

Inference Engine

The inference engine's reasoning processes are based on formal logic. In a simple production rule system the inference strategy is usuallly a variation of the modus ponens principle, which states that if

the conditions of a rule are satisfied, then it may be inferred that the result is also true. To illustrate, assume that a knowledge base contains the rule "If a person is a secretary then that person can type" and that the working memory contains the fact "Martha is a secretary". From this it is logical to conclude that Martha can type.

Systems which are built on logic programming languages use more sophisticated proof techniques based on the predicate calculus. These techniques are provided as standard control mechanisms in the language, much as DO-LOOPS are provided by FORTRAN.

Human experts often solve problems in situations where information is missing or uncertain. Consequently, inference engines should also be able to operate with incomplete data. To provide this capability most systems include facilities which allow users and knowledge engineers to tag facts and rules with certainty factors. Special combining rules enable the system to infer conclusions which are similarly qualified.

The control portion of an inference engine determines which rules will fire and the order of their firing. There are two basic control strategies: backward chaining, or goal directed, and forward chaining, or data driven. Both depend on the fact that the rules in a rule-based system form a hierarchical structure, with paths through the hierarchy from initial facts and conditions to conclusions.

Backward chaining is appropriate when a problem has a few well defined solutions (goal states). The system selects a tentative goal or hypothesis, perhaps based on some external priority, and examines its conditions. If working memory contains these conditions as facts, then the goal is established. Otherwise, the conditions will in turn be the goals of other rules and the system will try to establish them. By following this procedure repeatedly, the inferencing process can work backwards to the initial facts of the case. If the first goal selected cannot be proved, the system will select another and try again.

Forward chaining is a more complex procedure but is appropriate when there are many possible goal states, or when the goals are poorly or incompletely defined, as in planning situations. In this strategy

the rule hierarchy is traversed from initial facts to goals. The contents of working memory are compared to the condition clauses of rules and those rules whose conditions are satisfied are eligible to fire, thus modifying the contents of working memory. Repeated applications of this "recognize-act" cycle will eventually produce a solution.

An immediate consequence of the system architecture outlined above is that expert systems are relatively easy to modify. Because the knowledge base is completely separate from the control structure it can be developed incrementally. As knowledge in a domain increases or changes, rules can be added or modified. In fact, it is feasible to replace the entire knowledge base with one from another domain, since the inference engine is not problem dependent. This is the basic principle underlying the concept of expert system shells. The algorithmic structure of conventional programs, on the other hand, does not lend itself so readily to change. Knowledge and control are closely integrated in the program code and changes in one facet of the program frequently require changes in the other.

User Interface

One final aspect of expert system design is the user interface. Most expert systems are based on a consultation paradigm, where the user supplies the initial problem parameters in response to a series of questions proposed by the program. At certain points during execution the system may ask for additional information. Under ideal circumstances this dialogue would be conducted in natural language, but since natural language processing is not yet a mature technology it is more common to find restricted forms of communication based on menus or one word responses. A few systems are able to process English language responses within a very limited area, but this is the exception rather than the rule.

Explanation facilities are included in many expert systems. If the user wonders why a particular question was asked or how a conclusion was reached, he can interrogate the system directly. Explanations are usually little more than straightforward translations of the rules which are currently under consideration, and do not attempt to offer a deeper

rationale. Nevertheless, the ability of an expert system to justify its own actions is considered by many people to be a valuable feature, and is one of the characteristics that distinguishes expert systems from traditional programs.

2. APPROPRIATE PROBLEMS

Not every problem is a good candidate for expert system solution. Conventional computational techniques are still preferable in situations where clearly defined procedures, or algorithms, are available. Expert system technology should only be considered when one or more of the following conditions is present:

> The task to be performed is usually handled by an expert who uses heuristic rather than algorithmic techniques.
>
> The task is combinatorily explosive. There are so many conditions and complex interactions involved that even experienced human experts have difficulty considering all of them.
>
> The task is qualitative rather than quantitative, involving symbolic rather than numerical manipulations. Solutions may be subjective or judgmental in nature.

In addition to the above factors which characterize the problem domain, the current state of expert system technology imposes further restrictions. There must be at least one person, expert in the task domain, who is able to explain his expertise and problem solving techniques. The domain itself must be relatively narrow and well defined. Improvements in machine learning and knowledge acquisition methods may at some future date enable these restrictions to be relaxed.

Successful expert systems have been developed in a variety of fields, from medicine to engineering. An examination of the task domains reveals that the applications fall into a few broad categories, identified by function. A discussion of several of these categories will further serve to illustrate the range and applicability of expert system technology.

Diagnosis/Classification

Problems in this area try to identify a particular object or situation as being a specific instance from a set of possible situations. Identification is based on a group of "symptoms" provided by the user. Many diagnostic programs prescribe remedial action once the situation is identified. Typical applications include medical diagnosis and fault diagnosis in various kinds of systems.

MYCIN, developed as part of the Heuristic Programming Project at Stanford University, [Buchanan and Shortliffe, 84] is an excellent example of a medical diagnostic program. Its purpose is to identify bacterial infections and then prescribe an appropriate antibiotic treatment. This is a hard task, even for experienced diagnosticians. The physician must first decide if a significant infection is present and then identify the particular organism (or set of possible organisms) which is responsible for the infection. Finally, a suitable combination of drugs must be chosen to treat the disease. It is frequently necessary to make these decisions based on fuzzy and incomplete data; thus, good judgment is an essential component of the decision process.

MYCIN works as a backward chaining system. An initial hypothesis, based on preliminary patient data, is selected and guides the program in its consultation with the physician. Once the most likely organisms are identified, the program prescribes a drug treatment. Certainty factors attached to the action portion of rules enable the system to reproduce the judgmental behavior of domain experts.

DELTA (Diesel-Electric Locomotive Troubleshooting Aid) [Harmon and King, 85] was developed by the General Electric Company, Schenectedy, New York, to assist railroad personnel in the maintenance of diesel-electric locomotives. In addition to locating mechanical faults DELTA provides diagrams showing the faulty components and can, if requested, show training films which instruct maintenance personnel in the necessary repair procedures.

Planning

Planning problems are constructive in nature, unlike diagnostic programs, which have a fixed set of possible solutions built in. Plan generation may be

described briefly as follows: Given a task and the current situation, decide how to perform the task. [Charniak and McDermott, 85, p. 487] By decomposing the initial problem into subtasks, perhaps down through several levels, the system eventually identifies a set of elementary tasks whose solution plans are built into the system. It can then construct a master plan based on these subplans. Usually, constraints of various kinds must be incorporated into the planning process.

One of the best known planning programs is R1, sometimes known as XCON. P1 was developed by John McDermott [McDermott, 81]. It is used regularly by Digital Equipment Corporation to configure VAX computer systems. There are no standard VAX systems; instead, customers choose from a list of several hundred components which must then be arranged to conform to physical constraints imposed by the components themselves, the size of the area in which the system is to be installed, and other considerations. Due to the complexity of the systems and frequent modifications to the data base, the problem is difficult and time consuming even for experienced personnel. R1 has reduced system configuration time from hours to minutes, with a savings to the company which is measured in millions dollars.

The MOLGEN programs are equally successful in an entirely different domain. Their purpose is to design experiments for molecular geneticists to use for analyzing DNA molecules. Originally developed at Stanford University, MOLGEN has evolved into GENESIS, a package of several expert systems available from IntelliCorp (formerly IntelliGenetics). Users have the option of accessing GENESIS through a time sharing system or as a program to run on a LISP machine. [Harmon and King, 85]

Intelligent Instruction

Traditional computer-aided instruction consists of little more than an electronic textbook or training manual supplemented by question and answer drills. In contrast, intelligent assistance can be provided by embedding the knowledge bases of expert systems into larger programs. In addition to expert domain knowledge, intelligent tutors must contain teaching expertise and some way of interacting with the student.

GUIDON, developed by William Clancey at Stanford, was an early attempt to adapt an expert system for

instructional purposes. [Clancey, 82] It uses the knowledge base from MYCIN augmented by some two hundred additional rules which outline teaching strategies, methods of communicating with students, and explanation techniques. GUIDON has been used experimentally to train medical students at the Stanford School of Medicine.

Other instructional programs, notably STEAMER [Harmon and King, 85] and SOPHIE [Brown, 82], while not true expert systems, are still able to provide intelligent guidance to students by employing many of the techniques of knowledge engineering. STEAMER is being developed by the Naval Research Personnel Development Center in conjunction with Bolt Beranek and Newman. Its purpose is to train naval officers to run the steam propulsion plants in naval ships. SOPHIE tutors students in the art of troubleshooting electronic devices. Both programs use sophisticated simulation models of the systems being taught. Students are allowed to change the simulations interactively and observe what happens as results of the changes are propagated through the model.

Search

Occasionally expert systems are used in domains where algorithms are also available. Typically, these are "generate-and-test" algorithms. They provide a method of systematically ennumerating candidate solutions, each of which must then be tested to determine if it matches the problem statement. Expert systems use heuristics to limit the number of possible solutions generated, thus reducing significantly the amount of time spent in testing.

DENDRAL, developed at Stanford, is a good example of a heuristic search program. The task is to determine the structure of a particular molecule when given information about its component atoms and its mass spectra. [Barr and Feigenbaum, 82] Heuristics provided by expert chemists act as constraints, ruling out certain structural features and requiring the presence of others. The program is reported to perform at a level consistent with, or better than, domain experts.

This is by no means a comprehensive survey of expert system applications. As with any rapidly developing technology, new uses appear on the scene regularly. Some involve small systems which may not even deserve the name "expert". Nevertheless they are useful in many situations as an extension of or replacement for human involvement. Monitoring industrial equipment and processes is an example of this kind of application.

3. DEVELOPMENT AIDS

The earliest expert systems were developed using high level languages intended specifically for artificial intelligence applications, notably LISP. LISP provides a great deal of flexibility in the way information can be organized and is expressly designed to perform symbolic manipulations. Other AI languages such as PROLOG have also been used. Conventional programming languages (FORTRAN, for example) are oriented toward numerical processing and do not contain many of the features which simplify the job of the AI programmer.

In recent years the trend has been to provide even more assistance than is available from special purpose languages. Software tools developed expressly for expert system development are now being marketed for all size machines and in all price ranges. Most commercially available expert system tools are LISP-based, although some are written in PROLOG or PASCAL. Tools are in general less flexible than high level languages, but compensate for this loss of flexibility by offering a variety of special features which simplify the job of the knowledge engineer. Typically, these features would include an inference engine, various aids to knowledge acquisition and knowledge representation, and utilities to help with the development and debugging of the knowledge base.

Small expert system shells are designed to run on personal computers and support the development of systems containing a few hundred rules. Most of these tools are based on the diagnostic model, use backward chaining as the primary control strategy, and represent facts as O-A-V triples. Interaction with the user is through menus or a question-and-answer format. Little support is provided for knowledge engineering. Knowledge bases are typically created outside of the system using a word processor, although a few systems prompt for knowledge entry. Some provide trace features that let the user watch the rules as they fire. This is a useful method of debugging the knowledge base. Examples of this class of tools are ES/P ADVISOR (Expert Systems International), M.1 (Teknowledge, Inc.), and Personal Consultant (Texas Instruments).

Large system building tools are much more powerful than the simple aids discussed above. They run on large computers or special LISP workstations and can be used to

construct systems with several thousand rules. Instead of offering a single paradigm, they provide an array of features that can be used to customize a program to the application. System builders can choose from several different control strategies. Most common knowledge representation schemes are available, including O-A-V triples, frames, inheritance and certainty factors. Sophisticated graphics capabilities aid in debugging and the design of special screens. Such flexibility does not come without a price, however. The knowledge engineer must be skilled in the use of the tool and must be thoroughly familiar with the task domain in order to choose the most suitable methods out of the many that are offered. Software vendors usually provide workshops or on-site consulting to train programmers in the use of the tool. Examples of large expert system tools are ART (Automatic Reasoning Tool) from Inference Corp. and KEE (Knowledge Engineering Environment) from IntelliCorp.

Expert system developers may employ different tools at different stages. An initial prototype, which serves as a proof-of-concept, might be built quickly with one of the smaller expert system shells. The final prototype expands on this preliminary version by augmenting the knowledge base, providing graphic interfaces, and adding other features. A large tool is typically required at this point. For additional information on factors to consider in choosing a tool see Citrenbaum, Geissman and Schultz. Surveys of some of the current commercial tools are available from Harmon and King, and Gevarter. (See references)

4. LIMITATIONS

The advantages of expert systems have been widely advertised. They solve problems that have previously been intractable to computer solution, using programs that can evolve to keep pace with changes in the problem domain. They provide expert advice and assistance to an entire community of users, allowing human experts the freedom to work on new problems. They reduce computationally explosive tasks to a manageable size. The benefits of expert system technology should not, however, blind potential users to its limitations.

The heart of an expert system is its knowledge base and this is also the source of many of its limitations. Knowledge acquisition is a costly and time-consuming process, requiring months on the part of expert and knowledge

engineer alike. Once the knowledge base is built, the problem becomes one of truth maintenance. As rules are added over time, it becomes increasingly likely that contradictions and inconsistencies will be introduced. Size is also a problem; even modest systems may require many hundreds of rules. One practical consequence of all this is the necessity to restrict expert systems to narrow, well-understood task domains. Current technology does not support building the enormous knowledge bases that would be needed to handle problems with a broader scope.

Reliability is also of critical concern. Most systems employ a form of "shallow" reasoning; that is, inferences are based on heuristics and empirical data rather than being derived from domain axioms and principles. As a result, such systems tend to perform poorly when confronted with unexpected situations which were not anticipated in the original design.

Other problems are a result of the need to interface the system to humans, both during the building and operational phases of its lifetime. Ideally, one would like to be able to enter rules and describe problems in English, but in fact most systems still require the use of a stylized and sometimes obscure knowledge representation language.

Explanation facilities are limited in their ability to justify system behavior. A typical explanation consists of English translations of the rules involved in reaching that conclusion. Definitions of terms, causal relationships,and other potentially helpful information cannot be presented since it is not represented in the knowledge base.

An even more fundamental problem concerns the basic nature of expertise. Many skeptics question whether IF-THEN rules can fully capture the essence of an expert's know-how. [Dreyfus, 86] They claim that while rule based systems may be able to perform credibly in some situations, true expertise operates on many levels and cannot be reduced to an analytic process.

CONCLUSION

In the last five to ten years interest in artificial intelligence has increased dramatically. Established companies are developing AI departments, government agencies are investing enormous sums in AI research, and private software firms specializing in AI products are proliferating. Most of this activity centers on expert systems and related technology. Although there are still relatively few workable systems in use today, many prototypes are currently undergoing development and refinement.

It is reasonable to project that within the next five years small, special purpose expert systems will become common. Implemented on personal computers or as specially designed micro chips, these systems could perform reliably in many situations. Programs to monitor instruments and industrial machinery, to serve as intelligent procedure manuals or training aids, and to retrieve information from data bases are within the reach of current technology. Many of these small systems will be more appropriately called "competent" rather than "expert". When (and whether) large, truly expert systems will become widespread depends on the progress made by AI researchers in a number of areas.

First, there must be significant improvements in techniques of knowledge representation and knowledge acquisition. New methods for incorporating causality and domain principles into the reasoning process will enable systems to handle novel situations and to better justify their actions. Current research in computer learning techniques offers great promise for the future. A system which can learn domain rules by analyzing examples provided by an expert will greatly simplify the process of knowledge acquisition. Extrapolating even further, one can imagine a system provided with basic domain knowledge which will enable it to generate solutions by trial-and-error; and which, by observing the results, is able to derive its own rules independent of expert assistance. Methods of incorporating general world knowledge and common sense knowledge would provide even more human-like performance.

Currently there are ongoing research projects in all of these areas. Although advances have been made there is still much that remains to be done before significant

results will appear. There are some who think that order-of-magnitude improvements will not be possible unless new computer architectures are developed. They argue that today's sequential processing machines do not model the working of the human brain closely enough to successfully simulate intelligent behavior.

Regardless of whether expert systems are truly intelligent or can be considered accurate models of human expertise, it is becoming increasingly apparent that they can, in fact, be an important aid to computational problem solving. Existing systems have demonstrated the ability to perform reliably when used in an appropriate context and without unrealistic expectations. System designers who are aware of the weaknesses of expert system technology as well as of the advantages will be able to realize its full potential.

REFERENCES

Barr, Avron and Feigenbaum, Edward A., eds., "Applications-oriented AI Research: Science", in The Handbook of Artificial Intelligence, Vol. II, William Kaufmann, Inc., Los Altos, California (1982).

Brown, J. S., Burton, R., and deKleer, J., "Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III", in Intelligent Tutoring Systems, Sleeman, D. and Brown, J. S., eds., Academic Press, Cambridge, Mass., (1982).

Buchanan, Bruce G. and Shortliffe, Edward H., Rule Based Expert Systems, Addison-Wesley Publishing Company, Reading, Mass. (1984).

Charniak, E., and McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley Publishing Company, Reading, Mass., (1985).

Citrebaum, Ronald, Geissman, James R., and Schultz, Roger, "Selecting a Shell", AI Expert, Vol. 2, No. 9, 30-39, (1987).

Clancey, W. J. "Tutoring Rules for Guiding a Case Method Dialogue", Intelligent Tutoring Systems, Sleeman, D. and Brown, J. S., eds., Academic Press, Cambridge, Mass. (1982).

Davis, Randall, and King, Jonathan J., "The Origin of Rule-Based Systems in AI", in Rule Based Expert Systems, Buchanan, Bruce and Shortliffe, Edward H., Addison Wesley Publishing Co., Reading, Mass. (1984).

Dreyfus, Hubert and Stuart, "Why Expert Systems Do Not Exhibit Expertise", IEEE Expert, Vol. 1, No. 2, 86-90, (1986).

Gevarter, W.B., "The Nature and Evaluation of Commercial Expert System Building Tools", IEEE Computer, Vol. 20, No. 5, 24-41, (1987).

Harmon, Paul, and King, David, Expert Systems: Artificial Intelligence in Business, John Wiley and Sons, New York, NY (1985).

McDermott, J. "R1: The Formative Years", AI Magazine, Vol. 2, No. 2, 21-29 (1981).