Behavior Research Methods, Instruments, & Computers
1989, 21 (2), 195-204

SESSION VII
TUTORIAL ON EXPERT SYSTEMS

Doris Aaronson, *Presider*
New York University

Expert systems: A cognitive science perspective

ROY LACHMAN
University of Houston, Houston, Texas

The theory and technology of knowledge-based systems are intrinsically interdisciplinary and are closely related to the formalisms of cognitive psychology. In this paper, strategies of incorporating intelligence into a computer program are described along with a common architecture for expert systems, including choices of representation and inferential methods. The history of the field is traced from its origins in metamathematics and Newell and Simon's (1961) GENERAL PROBLEM SOLVER to the Stanford Heuristic Programming Project that produced DENDRAL and MYCIN (Buchanan & Shortliffe, 1984). MYCIN gave rise to EMYCIN and a shell technology that has radically reduced the development time and cost of expert systems. Methodology and concepts are illustrated by transactions with a shell developed for graduate education and a demonstration knowledge base for the diagnosis of senile dementia. Knowledge-based systems and conventional programs are compared with respect to formalisms employed, applications, program characteristics, procedures supplied by the development environment, consistency, certainty, flexibility, and programmer's viewpoint. The technology raises basic questions for cognitive psychology concerning knowledge and expertise.

The purpose of this paper is to explore the properties of knowledge-based systems and to compare them to conventional software systems. Such an analytic comparison requires an account of the conceptual origins of expert systems, their development, basic architecture, and relationship to psychological theory. The reader is assumed to have some familiarity with programming in a high-level language and at least an elementary sense of basic data structures and other programming concepts. This appears to be the characteristic level of awareness of most research psychologists, whose major effort is naturally devoted to the factual content and methodology of their primary occupation—the research and teaching of psychology. My examination of expert systems will follow a top-down sequence of exposition. The relationship of expert systems to cognitive science will be discussed first, followed by an examination of central issues, such as what it means to incorporate intelligence and knowledge into software systems, the architecture of intelligent systems, and their control processes. The discussion will focus on the MYCIN system, and on its predecessor and progeny, since those programs have produced major advances in theory and technology. The final sections contain examples of transactions with an expert system used for demonstration and teaching, and an analysis and comparison of expert and conventional system functionality. The paper, in its entirety, is based on the premise that basic knowledge will be advanced and the effective use of expert systems in science will be facilitated by a careful distinction between conventional computer-support systems and knowledge-based systems.

Address correspondence to the author at the Department of Psychology, University of Houston, Houston, TX 77204-5341.

EXPERT SYSTEMS AND COGNITIVE SCIENCE

Expert systems are one type of knowledge-based system and as such unequivocally comprise a subfield of artificial intelligence (AI), which itself is a division of computer science. However, the designers of expert systems rely on the factual content and methodologies of several autonomous disciplines. These are collectively called cognitive science, and include linguistics, cognitive psychology, and AI. These disciplines share in common the study of various kinds of operations on symbols and symbol systems. Cognitive scientists tend to be interested in the nature and properties of intelligent systems, and seek, in

varying degrees, computational theories of the systems of interest to their discipline. The cognitive sciences differ, as do all disciplines, in their presuppositional structures; they have different conceptions of subject matter, the appropriateness of specific problems, the acceptability of solutions, and the concepts of proof. The different presuppositions concerning the nature of truth and the methods of proof (Lachman & Lachman, 1986) result in the selective use of confirmation methodologies, such as the natural science model of experimentation, formal proofs, intuitions of native speakers, and proof by demonstration of a running program. The differences are overshadowed, however, by the interdependence of the cognitive sciences in finding solutions for each discipline's sanctioned problems. In particular, several of the core problems of knowledge-based systems are indistinguishable from the central problems of cognitive psychology and may require the methodology of both disciplines for acceptable solutions. Finally, it should be emphasized that although conventional software can be explicated in the idiom of computer science, knowledge-based systems can only be understood in terms of the concepts and methods of several of the individual cognitive sciences. The standard architectures of knowledge-based systems, discussed below, clearly show that dependence.

INCORPORATING INTELLIGENCE IN A COMPUTER PROGRAM

Traditional programs are rigidly organized procedures for performing a task or solving a problem interactively with the user. The procedures, developed in software-engineering environments, are coded into programs from specifications provided by domain experts, project directors, or analysts, all of whom tend to have naive models of end-users. The experts that develop specifications for a program possess explicit knowledge of the problem area and base their software requirements on established problem-solving methods. Conventional programs are then constructed from those specifications. The program code represents algorithms, that is, step-by-step symbol-manipulating operations, including decision and branch points, that are infallible for obtaining some result or solving a problem. In some problems, for which there are a limited number of potential solutions, the algorithm consists of an exhaustive search through all possibilities. However, many significant scientific and human problems cannot be solved by algorithms because the problem areas are ill defined or open-ended, or because an exhaustive search may lead to combinatorial explosion. In such a situation, there are too many solution paths to search in a reasonable amount of time.

AI-based expert systems, in contrast, rely on heuristic methods to limit the search space, or number of solution paths, for a problem. Heuristics are procedures based on the established facts of a field and the judgment, opinions, and intuitions of a domain expert. Expert systems explicitly represent, in computer code, the knowledge that makes decisions possible. They can thus function with poorly specified problems just as the expert does; they are not confined to the precisely specified problems that are the only type typically solved by conventional software.

A major difference between expert systems and conventional programs is that the capabilities of conventional programs come largely from stored algorithms making up their procedures, whereas the intelligence in expert systems is derived primarily from the explicit knowledge stored in its knowledge base. The knowledge there may be coded as rules, frames, or semantic networks. However it is coded, the declarative knowledge representation is processed deductively by a family of algorithms variously called the inference engine or interpreter. Conventional programs, on the other hand, must be constructed in a thorough, precise, and exhaustively specified fashion or they will not run. They are rigid and absolutely intolerant of ambiguity; failure of the user to conform to a program's requirements explicitly, including hidden requirements, leads a conventional program to abort or "hang up." Gaps in knowledge, therefore, are not tolerated. In contrast, knowledge-based software systems make no such demands, basically because intelligence is incorporated into the system in a segregated, declarative knowledge base. The knowledge, semantics, and intelligence of conventional programs are embedded in algorithms and distributed throughout combinations of procedures. This makes them context-dependent, difficult to understand and to modify, and highly predictable. The intelligence in knowledge-based systems, on the other hand, is primarily in the declarative expression of rules, semantic nets, or frames. The knowledge is recorded as data, is context-independent, and is easy to understand and modify. Such systems are rather unpredictable, because part of their intelligence is in the semantic routines that control the state of the system, and the consequences of that control can be very difficult to anticipate.

The relative power of the two approaches for various kinds of scientific and practical problems is very different. Articles on psychology and computing, such as those dealing with computer-based decision aids, sometimes fail to distinguish between intelligent and conventional programs. Indeed, at one level of analysis there is considerable overlap between expert systems and standard programs. Expert systems, depending on their design, may contain various structures and procedures that are also present in conventional programs, and the algorithms of standard programs do represent human knowledge. Moreover, both types of systems run on coded formalisms that are *Turing-complete* (Minsky, 1967). A Turing-complete formalism coded into a procedure is one that can be executed on a Turing machine, and hence on any instantiation of a Turing machine, including high-level computer languages. Thus, anything that can be coded in the formalisms of expert systems can also be coded in conventional high-level computer languages, given enough programming skill, coding time, and patience (Haugeland, 1981). Nevertheless, software systems that contain raw

expertise in a knowledge base are crucially different from those based solely on algorithmic methods.

THE ARCHITECTURE OF EXPERT SYSTEMS

A software system is one or more configurations of computer code (a particular program, a programming language, or a programming environment that consists of one or more languages and numerous utility programs). Various representations of software systems differ in level of abstraction and amount of detail. Machine language is the most concrete level of software system and pseudocode is the least. Pseudocode is a rough natural-language representation of the anticipated steps in a program. It tends to be very loose and tentative, and lacks detailed expressions of implementation. The level of description called the architecture is less abstract than pseudocode, and has greater detail. However, representation at the architecture level is more abstract and less detailed than any level of computer code. The architecture of a software system, therefore, represents the system at a level of abstraction intended to capture the functionality and principles of operation built into the system, but is uncluttered by input/output and other details of computational procedure. Architecture, presented in a block diagram, is a simplification and idealization of a system; the diagram is designed to show functional components and their interconnections, and to capture major properties of the system and render them comprehensible.

The User Interface

A common architecture for expert systems is presented in Figure 1. The natural-language interface represents software elements that control the system's output of queries and conclusions, and its use of input data from the user regarding problem states and parameter values. Typically, the screen display is the main output modality and the keyboard is the input modality of choice. Design of the user interface is a very active area of psychological research (Carroll, 1987). Interface designs, with very few exceptions, are rendered in algorithmic programs. Effective heuristic design of the user interface, based on natural-language communication, requires cognitive science (linguistics, psycholinguistics, and perhaps even ethnomethodology). In expert systems, the user interface solicits facts from the user about the current problem, in the form of multiple-choice selections or open-ended questions with a limited set of acceptable responses. The output information presented to the user, although it often appears to have many features of generative natural language, is usually "canned." That is, the system developer has previously constructed, in the vernacular of the knowledge domain, natural-language sentences that describe each possible problem solution and the reasons that a given solution was selected by the system. Although real progress has been made in computational linguistics, cognitive science and AI technology have a long way to go before a truly intelligent natural-language interface will be available (Winograd, 1982).

The Knowledge Base and Knowledge Representation

The components of the architecture in Figure 1 are abstractions. The knowledge base, for example, is a conceptual receptacle for real-world knowledge. Knowledge in the base is stored in a declarative form as data structures that may or may not contain procedural elements. During the construction of an expert system, knowledge, in the form of rules, frames, and semantic nets, can be entered in various ways, depending on the development environment. Knowledge is entered into the base either by coding in a high-level language such as LISP or by use of a *shell*, which is a particular type of expert-system software-development environment. There are many strategies for representing knowledge, and cognitive psychol-

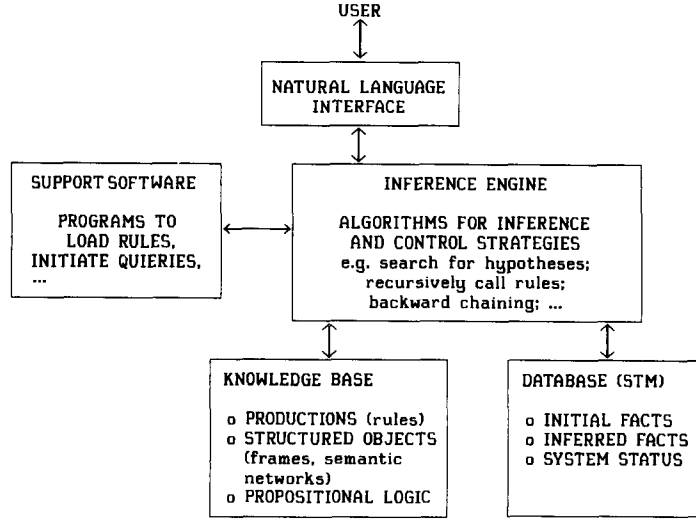

Figure 1. A common architecture for knowledge-based systems.

ogists tend to be familiar with one or more of the approaches, so we will not linger on the important topic of representation. Using Nilsson's (1980) classification, there are three major classes of representation: production systems (also called rule-based systems), logic systems (e.g., predicate calculus), and structured objects (e.g., semantic nets and frames). A fourth strategy is increasingly being used; it consists of various hybrid combinations of representational formats. Each category of representation has subtypes, and the pure versions of the major types have been proven to be formally equivalent (Anderson, 1976; Minsky, 1967). Although the formal equivalence of representational systems is of considerable theoretical importance, it has limited consequences for the actual development of expert systems. During the historical development of knowledge-based systems, a strategy emerged that was quite successful in advancing AI science and technology. Declarative knowledge was conceptually and physically segregated, as much as possible, in the knowledge base, and procedural knowledge was identified with the inference engine. Knowledge representation in current expert systems tends to be declarative, but, as we have earlier noted, procedural knowledge sneaks in, in various guises (Buchanan & Shortliffe, 1984).

Inference Engine and Database

The inference engine represents a set of algorithms that produce inferences from declarative knowledge stored as rules (alias: production system), frames (alias: schema), semantic networks (alias: acyclic and directed graphs), or some combination of these representational formats. These algorithms control the "reasoning" process by combining the inferential schema implicit in rules, frames, and networks with characteristics of the current problem either obtained from the user or inferred from facts stored in the database. When an advisory session between expert system and user is initiated, the inference engine determines what facts it needs to solve a problem, and either gets those facts or terminates the computer run. The algorithms in the inference engine also control the sequencing and generation of inferences, add newly inferred facts to the database, and process confidence levels. Although the classic formalisms for generating inferences with productions and networks are Turing-complete and therefore formally equivalent, the actual algorithms implemented differ from the classical form of a production system or graph. Modifications to the formalism are made for pragmatic design considerations. Inferential algorithms for each representational schema can be implemented in many different ways with different selections of features and, therefore, really represent distinguishable families of procedures associated with productions, frames, networks, and propositional logic.

Algorithms for making deductions from uncertain and incomplete information are often included in the inference engine. Both knowledge-based and standard programs work in a deterministic fashion when evaluating probabilities of events, but the user of a standard program usually supplies all the data, estimates missing data, or selects a rule for making that estimate. Expert systems, in contrast, are characteristically designed to handle missing and uncertain data automatically. It is the forte of expert systems to work with uncertain, incomplete, and fuzzy data.

In productions (also known as conditionals or rule-based inferences), the simplest and most basic rule of inference is *modus ponens*. The schema is:

If p then q; given p; infer q

The conditionals are recorded in the knowledge base and the initial facts about the problem are obtained from the user and stored in the database. Algorithms of the inference engine apply the facts as antecedents of the conditionals and derive new facts from the consequence which are then also stored in the database. The system searches through the database, queries the user, or does both in an effort to find new antecedents so that rules may be fired. Firing a rule means finding a pattern that matches p in the rule base so that q may be added to the database as a new pattern and therefore as a potential match to forthcoming occurrences of p. The process requires a starting point, a strategy for ordering the selection of rules, and a strategy for working either from facts to conclusions, from conclusions (or hypotheses) to supporting facts, or both ways. The former strategy is called "forward chaining" and the latter strategy "backward chaining"; both strategies originated in mathematical logic.

The main menu of a shell that supports the development of simple expert systems is shown in Figure 2. Computer runs showing the output from the shell's algorithms that represent the forward- and backward-chaining strategies for an identical rule base called DEMENTIA.RB are presented in Figures 3 and 4, respectively. The computer transactions shown in the figures are for DEMENTIA rules, three of which are shown in Table 1. The forward-chaining algorithm, depending on its particular implementation, elicits the initial problem parameters (i.e., the facts) from the user and proceeds to draw all possible conclusions, as is illustrated in Figure 3. It recursively calls the rules by searching the database for an antecedent and adding the consequences of fired rules to the database until all facts and rules are exhausted. The forward-chaining algorithm can be implemented in a conversational expert system or in a larger system with primarily conventional components. The forward-chaining strategy is followed when it is possible to assume that all of the required information will be available at the start of a computer run. The backward-chaining algorithm can be designed to search the rule base recursively until it produces a set of terminal nodes or end-states such as the hypotheses in Figure 4. Backward-chaining and hybrid systems will tend to be conversational. Selection of a hypothesis is solicited from the user, and the necessary facts to prove or discard it are obtained interactively, as in Figure 4. The backward-chaining sequence establishes subgoals and treats them in a similar fashion as the original hypothe-

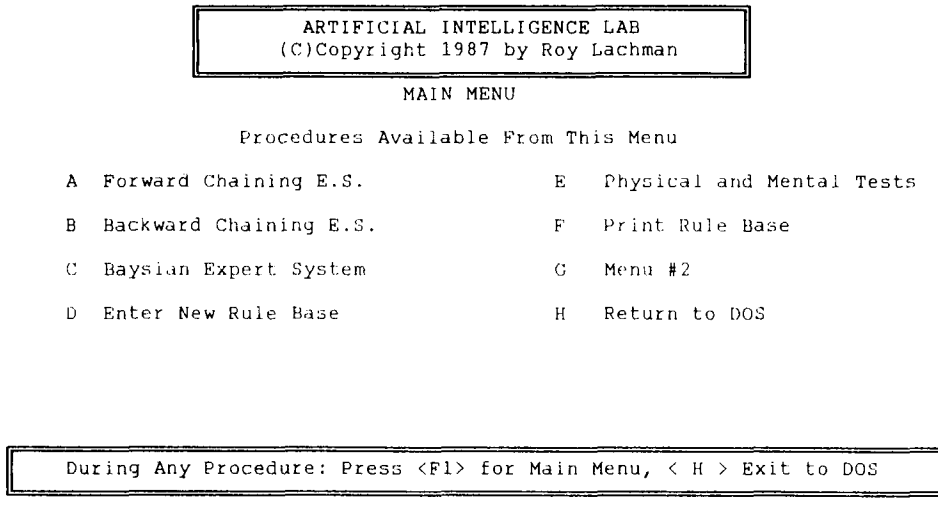

Figure 2. The main menu of an expert-system shell used for graduate training in the cognitive theory and technology of knowledge-based systems.

sis. Subgoals are evaluated by testing the first rule and, if it fails to fire or to establish the subgoal, recursing on the remaining rules. Subgoals that are established as facts (i.e., are the consequence of a fired rule) are added to the database. The backward-chaining algorithm either establishes the selected hypothesis, or if it cannot, it says so and terminates the computer run.

Support Software

Support software, in an expert-system development environment, performs any of a number of utility and support functions. These functions can deal with any aspect of the architecture. In a rule-based system, for example, procedures can be included that interactively construct rules, edit the rules, test them for syntax and internal consistency, or test the entire rule set for contradictions. Graphics capabilities, development programs, file location and reformatting, and the like are available for other components of the architecture. In fact, any procedures used in user-interface development environments or in general software engineering can be added to an expert-system shell.

ORIGINS OF THE KNOWLEDGE-BASED SYSTEM ARCHITECTURE

Although a number of intellectual achievements made the emergence of AI and cognitive psychology possible, the seminal work in metamathematics in the 1930s ultimately contributed to both. The joint origins of AI and cognitive psychology in mathematical logic is described from the cognitive perspective by Lachman and Lachman (1986), and from the perspective of AI by Newell and Simon (1976). A direct line of development can be traced from the publication of Godel's (1931/1965) incompleteness theorem and Turing's (1936) computability thesis to the physical symbol system hypothesis (Newell, 1980; Newell & Simon, 1976). Although the physical symbol system hypothesis was not explicitly expressed until the 1970s, it is implicit in the early landmark work of Newell and Simon described below. The production system formalism, so important to contemporary psychological theory and AI, also originated in the metamathematics of the 1930s (Post, 1936).

In the 1950s, Newell and Simon originated a research program, in the Lakatosian sense (Lakatos, 1970), that was to become a cornerstone of psychology's information-processing paradigm. The program had several goals. One was to understand and develop computer programs that could deal with ultracomplicated problems such as chess or theorem proving in mathematical logic. Another goal was the scientific explication of general problem-solving and decision-making methods that involved limited rationality, "satisficing," and selective search. The computer programs they developed in that connection, LOGIC THEORIST (Newell & Simon, 1956) and GENERAL PROBLEM SOLVER (GPS; Newell & Simon, 1961), are illustrative of both the main research emphasis and the kinds of theory that emerged from their laboratory. Their research program aimed at the elucidation of domain-independent general problem solving and the development of theories of human thought. Their psychological theory was instantiated in a series of running programs whose performance generally was comparable to that of human problem solvers (Newell & Simon, 1972). The AI objective of the program was to achieve general intelligence and domain-independent problem solving in a heuristic machine. Students of Newell and Simon wrote theses and dissertations, and conducted postdoctoral research using, and in some cases extending, the approach enunciated in GPS. While preparing his dissertation under the tutelage of Simon, Feigenbaum worked within the prevailing paradigm and developed a simulation of verbal learning called EPAM (Feigenbaum, 1961). However, attempts to develop programs in that tradition as serious technology ran into insurmountable problems.

```
02-16-1988    20:21:17                    RULE BASE=DEMENTIA.RB

FORWARD CHAINING INFERENCE ENGINE

To control the screen output Press <C>, otherwise <ANY KEY>.

Enter facts now - Enter * To Stop.
Follow each entry with <RETURN>.

FACT? NOT SELF-CARE
FACT? POOR-RECENT-MEMORY
FACT? NOT COUNT
FACT? MAINTAIN-CONVERSATION
FACT? *


STM (data base) contains:
NOT SELF-CARE POOR-RECENT-MEMORY NOT COUNT MAINTAIN-CONVERSATION

The system is trying to find a fact in STM and to
match it to an applicable rule in the rule base.


SYSTEM USING RULE  17
Rule  17  Deduces: SEVERE-DECLINE
STM (data base) now contains:
NOT SELF-CARE POOR-RECENT-MEMORY NOT COUNT MAINTAIN-CONVERSATION
SEVERE-DECLINE


SYSTEM USING RULE  3
Rule  3  Deduces: EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)
STM (data base) now contains:
NOT SELF-CARE POOR-RECENT-MEMORY NOT COUNT MAINTAIN-CONVERSATION
SEVERE-DECLINE EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)

NO MORE APPLICABLE RULES
FINAL RESULT: EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)


02-16-1988    20:22:00 NORMAL TERMINATION
```

Figure 3. Computer transactions for forward chaining on the DEMENTIA rule base. All user entries are made to the prompt "FACT?". The domain of expertise used to illustrate these principles and methods is the differential diagnosis of senile dementia.

Feigenbaum, in collaboration with the geneticist Lederberg, first attempted to develop a comprehensive and domain-independent aid to scientific thinking. Their effort was not fruitful until they focused the problem-solving heuristics of the project on a circumscribed domain, and abandoned their effort to construct a general problem-solving program. The resulting system of programs, called DENDRAL, was developed by Feigenbaum, Lederberg, Dejerassi, and others. DENDRAL is a forward-chaining system that describes the molecular structure of unknown organic compounds from mass spectrometer and nuclear magnetic response data. It contains productions for the data-driven components and a procedural representation for the molecular structure generator (Lindsay, Buchanan, Feigenbaum, & Lederberg, 1980). DENDRAL was a watershed in AI; it represented a shift from power-based to knowledge-based programming. The program became the forerunner of many subsequent expert-system projects, in particular MYCIN. Success in the construction of knowledge-based programs thus was achieved only when the AI design objectives were changed to the representation of narrow, limited, and focused domains of expertise (Feigenbaum, Buchanan, & Lederberg, 1971).

The major change of focus from general, domain-independent to domain-dependent problem solving, along with other lessons learned from DENDRAL, were incorporated into the MYCIN family of programs of the Stanford Heuristic Programming Project (Buchanan & Shortliffe, 1984). Newell (1984) described MYCIN as the expert system "that launched the field." The development of MYCIN and related programs (see Figure 5) made explicit many of the issues that arise in the construction of knowledge-based systems. Although considerable innovation in AI technique has been achieved and landmark systems have been developed, numerous basic and applied problems in cognitive science that are related to the basic properties of knowledge and to characteristics of knowledge-based systems remain unsolved. The MYCIN projects, including their domains, epistemological properties, and development, provide what is likely the most informative perspective on expert systems. The continuing retrospective study of MYCIN (Buchanan & Shortliffe, 1984; Clancey, 1985) explains the basis of the rise to prominence of knowledge-based systems, the factors that make them successful, and the scientific foundations upon which they are built.

```
02-17-1988    14:40:08   VER=S14            RULE-BASE=DEMENTIA.RB

Try To establish one of the following hypotheses from the rule base:

 1 =NORMAL-AGING
 2 =BENIGN-SENESCENT-COGNITIVE-DEFICITS
 3 =EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)
 4 =MIDDLE-SENILE-DEMENTIA-ALZHEIMER-TYPE
 5 =LATE-SENILE-DEMENTIA-ALZHEIMER-TYPE
 6 =MULTI-INFARCT-DEMENTIA
 7 =POTENTIALLY-REVERSIBLE-DEMENTIA

Enter hypothesis No.  3
The System is trying to prove goal EARLY-SENILE-DEMENTIA-
ALZHEIMER-TYPE-(SDAT)
The System is trying to prove goal SEVERE-DECLINE
The System is trying to prove goal POOR-RECENT-MEMORY

All relevant rules have fired, can't prove POOR-RECENT-MEMORY,
which is not in STM-DB nor the consequent of a Rule.

IS THIS TRUE: POOR-RECENT-MEMORY
     Y=Yes
     N=No
     D=Don't know (Not yet operational)
     W=Why (Show contents of STM data base)

RESPONSE: Y

The System is trying to prove goal NOT COUNT

All relevant rules have fired, can't prove NOT COUNT,
which is not in STM-DB nor the consequent of a Rule.

IS THIS TRUE: NOT COUNT
     Y=Yes
     N=No
     D=Don't know (Not yet operational)
     W=Why (Show contents of STM data base)

RESPONSE: Y

The System is trying to prove goal MAINTAIN-CONVERSATION
All relevant rules have fired, can't prove MAINTAIN-CONVERSATION,
which is not in STM-DB nor the consequent of a Rule.


IS THIS TRUE: MAINTAIN-CONVERSATION

     Y=Yes
     N=No
     D=Don't know (Not yet operational)
     W=Why (Show contents of STM data base)

RESPONSE: Y

RULE  17 Deduces SEVERE-DECLINE

Short-term memory (DATA BASE) now contains:
POOR-RECENT-MEMORY NOT COUNT MAINTAIN-CONVERSATION SEVERE-DECLINE

The System is trying to prove goal NOT SELF-CARE
All relevant rules have fired, can't prove NOT SELF-CARE,
which is not in STM-DB nor the consequent of a Rule.

IS THIS TRUE: NOT SELF-CARE

     Y=Yes
     N=No
     D=Don't know (Not yet operational)
     W=Why (Show contents of STM data base)

RESPONSE: N

UNABLE TO PROVE HYPOTHESIS: EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-
(SDAT)

02-17-1988    14:41:04 NORMAL TERMINATION

                         o
                         o
                         o


The System is trying to prove goal NOT SELF-CARE
All relevant rules have fired, can't prove NOT SELF-CARE,
which is not in STM-DB nor the consequent of a Rule.

IS THIS TRUE: NOT SELF-CARE

     Y=Yes
     N=No
     D=Don't know (Not yet operational)
     W=Why (Show contents of STM data base)

RESPONSE: Y

RULE  3 Deduces EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)
Short-term memory (DATA BASE) now contains:
POOR-RECENT-MEMORY NOT COUNT MAINTAIN-CONVERSATION SEVERE-DECLINE
NOT SELF-CARE EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)

FINAL RESULT: EARLY-SENILE-DEMENTIA-ALZHEIMER-TYPE-(SDAT)

02-17-1988    15:04:57 NORMAL TERMINATION
```

Figure 4. Computer transactions for backward chaining on the DEMENTIA rule base. All user entries are made to the prompt "RESPONSE:". Entries are limited to the numbers 1 to 7 indicating the choice of hypothesis, and to the letters "Y" for Yes, "N" for No, and "W" for Why.

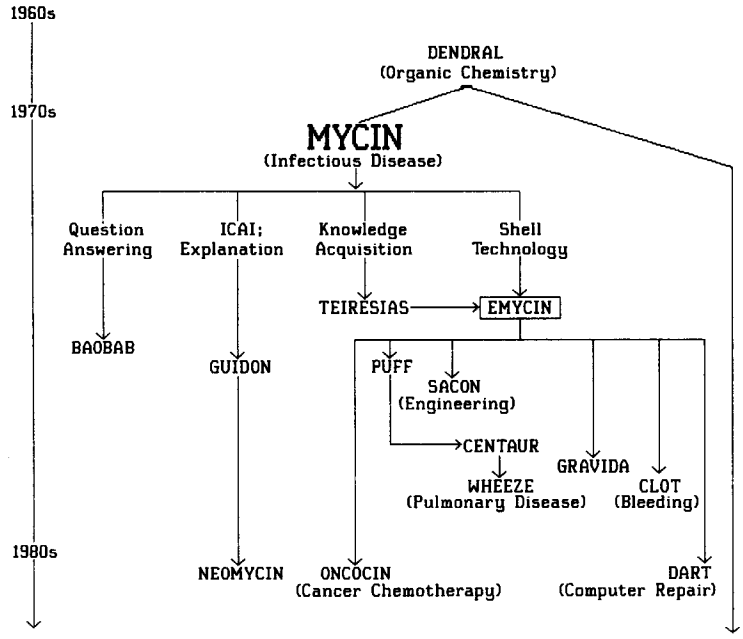

Figure 5. Program developed in the Stanford Heuristic Programming Project (Buchanan & Shortliffe, 1984).

The initial objective of the MYCIN project was the development of a computer-based consultation system for the diagnosis of infectious disease and for giving advice in the choice of antimicrobial therapy. In the case of acute infections, immediate treatment is necessary even before the results of all laboratory tests and cultures are available. In addition, antimicrobial therapy is subject to considerable human error. Consequently, development of the MYCIN system had practical significance, in addition to the scientific importance of testing AI constructs. A modified production system approach with a backward-chaining strategy was selected for the program. In other words, the system works backward from the goal of a therapeutic regimen and tests various rule-based diagnoses against facts supplied by the user interactively. The success of a rule-based approach in the DENDRAL programs was one reason for selecting a production-system representation. Still more important was the discovery by the developers of MYCIN that rules, and lines of reasoning based on chained rules, are easier to understand, critique, and modify than are alternative representations. These characteristics made fast prototyping possible with rule-based systems, an important property of production-system-based representation. The separation of production rules from the inference algorithms made it possible to develop an explanation module. The advice offered in a consultation was explained by the system, at least in part, by tracing the chains of inferences, displaying a description of the rules invoked, and indicating why the system requested certain pieces of information from the user. The developers discovered, early in the project, that the productions of MYCIN differed in a significant way from those of DENDRAL, and that many of the inferences produced in MYCIN were uncertain. A system was therefore developed to handle probabilistic components of rules and to combine separate elements of evidence for diagnostic hypotheses (Buchanan & Shortliffe, 1975).

The MYCIN system soon became the model for investigating a number of fundamental problems in AI and for developing expert system tools. The family of AI research programs was part of the Stanford Heuristic Programming Project shown in Figure 5 (Buchanan & Shortliffe, 1984). Research was conducted and programs were developed for expert system programming environments, on-line knowledge acquisition, tutorial packages, and consultation programs for a variety of domains, some of which are shown in Figure 5. The specific programs developed included inference, natural-language question answering

Table 1
Three Rules from the DEMENTIA Rule Base Diagnosing Levels of Alzheimer's Disease, Multi-Infarct Dementia, and Depressive Dementia

Rule 3	If	Severe-Decline
	And	Not Self-care
	Then	Early-senile-dementia-Alzheimer-type (SDAT)
Rule 6	If	Sudden-onset-cog-Decline
	And	Fluctuating-course
	And	Impaired-consciousness
	And	Vascular-disease
	Then	Multi-infarct-dementia
Rule 32	If	Poor-recent-memory
	And	Not count
	And	Maintain-Conversation
	Then	Severe-Decline

Note—DEMENTIA.RB was constructed from expert opinions published in Pearce (1984), Shamoian (1984), and other sources. The rule base is used only for training in the cognitive theory and technology of rule-based systems.

(Bonnet, 1980), intelligent tutoring (Clancey, 1986), computer-based knowledge acquisition (Davis, 1979), and the expert-system shell EMYCIN (van Melle, Shortliffe, & Buchanan, 1984). EMYCIN is of practical importance because it produced a technology that has been duplicated and extended in numerous commercial packages. The shell was created by removing the medical productions from the rule base of MYCIN and adding other procedures to the rule base. This made it possible to formulate rules from a variety of domains, including medical, engineering, and commercial, and to build a working system at a considerable reduction in time and cost. The shell developed in my laboratory and its output for DEMENTIA.RB shown in Figures 2, 3, and 4 were patterned on EMYCIN.

Many additional scientific lessons were learned from MYCIN and its progeny, some of which are validated by the demonstration of the expert-level performance of the system. The MYCIN project demonstrated that significant modifications to a program can be accomplished, without any reprogramming, by changing or adding declarative statements. From the origin of AI as a discipline, the capacity to improve performance without reprogramming has been viewed as a significant dimension of machine intelligence (McCarthy, 1958). The ability of the system to use the same knowledge in more than one way (e.g., for advising and for explaining) is also symptomatic of intelligence. An important lesson for cognitive psychologists is the repeated demonstration, through progeny of MYCIN, that a few hundred rules may suffice to represent the deep knowledge of various significant domains. Perhaps the final lesson of the project is the discovery of the limits of rule-based systems, of the kinds of problems for which they do not work well or at all (e.g., continuous monitoring tasks, such as monitoring of life support systems in intensive care medical facilities).

COMPARING KNOWLEDGE-BASED SYSTEMS WITH CONVENTIONAL SOFTWARE

The distinction between conventional and knowledge-based systems is both real and important. However, this distinction cannot be appreciated simply by reference to the basic formalisms that support either of the systems. There are several reasons that formal comparisons are problematic. First, experimental systems, such as MYCIN, explore variations in many aspects of a formalism, as well as in combinations of formalisms; this makes it unclear exactly what is being compared. Second, the previous description of procedural languages as being Turing-complete means that anything that can be computed with a production system can also be computed with a high-level procedural language (Haugeland, 1981). Thus, at the formal level of description, there is no distinction to be made between any of the Turing-complete computational systems. In physics and elsewhere, vastly different ontological domains can be represented by an identical mathematical formalism (Lachman, 1960). Consequently, although categories such as knowledge-based systems and conventional programs are identical at one level of representation, they are vastly different at another. There-

Table 2
Programming Environments

	Standard Programming System	Knowledge-based System
Applications	Operates in well-defined domains with well-defined problems	Operates in poorly defined domains with poorly structured problems
Flexibility	Literal, inflexible	Flexible
Certainty	Algorithmic, guaranteed to be correct	Not guaranteed correct, but can reach performance level better than human expert
Consistency	Requires absolute consistency	Deals with some inconsistency
Environment supplies	Iteration, branching, subroutine calls, etc.	Forward and backward reasoning, traces its decisions, recursive calling of rules, etc.
Program	A description of a set of calculations or formal operations	A description of relationships between variables, objects, or a system and its components
Control flow	Decision of process currently executing	Select-execute loop, unity of data and control
Viewpoint	Calculations performed, procedures implemented	Knowledge or rules applied, expanded view of a program
Programmer's job	Describe algorithms needed to solve problem	Select knowledge needed to solve problem

fore, both categories must ultimately be described, not only in formal terms, but in terms of their presuppositions and in their style of approach. Newell (1984) has observed that many kinds of conventional software are designated as expert systems, which "mongrelizes" the field of AI-based knowledge systems. It is conceptually worthwhile to distinguish between the two categories, and major differences have been described throughout this paper. A summary of the features of each approach is presented in Table 2. Several of the differences are, as previously described, only a matter of degree. However, the categories of Table 2 represent different perspectives on the concepts employed, the external systems modeled, the program, and the programmer. Taken together, the features of knowledge-based systems may alter human intellectual capacity to such a degree that both science and society are fundamentally changed.

REFERENCES

ANDERSON, J. R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Erlbaum.

BONNET, A. (1980). *Analyse de textes au moyen d'une grammaire semantique et de schémas. Application à la compréhension de résumés médicaux en langage naturel.* Thèse d'état, Université Paris VI, Paris, France.

BUCHANAN, B. G., & SHORTLIFFE, E. H. (1975). A model of inexact reasoning in medicine. *Mathematical Biosciences*, **23**, 351-379.

BUCHANAN, B. G., & SHORTLIFFE, E. H. (Eds.) (1984). *Rule-based expert systems*. Reading, MA: Addison-Wesley.

CARROLL, J. M. (Ed.) (1987). *Interfacing thought: Cognitive aspects of human computer interaction*. Cambridge, MA: MIT Press.

CLANCEY, W. J. (1985). Heuristic classification. *Artificial Intelligence*, **27**, 1-67.

CLANCEY, W. J. (1986). From GUIDON to NEOMYCIN and HERACLES in twenty short lessons: ONR final report 1979-1985. *AI Magazine*, **7**, 40-60.

DAVIS, R. (1979). Interactive transfer of expertise. *Artificial Intelligence*, **12**, 121-157.

FEIGENBAUM, E. A. (1961). The simulation of verbal learning behavior. *Proceedings of the Western Joint Computer Conference*, **19**, 121-132.

FEIGENBAUM, E. A., BUCHANAN, B. G., & LEDERBERG, J. (1971). On generality and problem solving: A case study involving the DENDRAL program. In B. Meltzer & D. Michie (Eds.), *Machine intelligence 6* (pp. 165-190). New York: Elsevier.

GODEL, K. (1965). *The undecidable: Basic papers on undecidable propositions, unsolvable problems and computable functions* (M. Davis, Ed. & Trans.). Hewlitt, NY: Raven Press. (Original work published 1931)

HAUGELAND, J. (1981). Semantic engines: An introduction to mind design. In J. Haugeland (Ed.), *Mind design* (pp. 1-34). Cambridge, MA: MIT Press.

LACHMAN, R. (1960). The model in theory construction. *Psychological Review*, **67**, 113-129.

LACHMAN, R., & LACHMAN, J. L. (1986). Information processing psychology: Origins and extensions. In R. E. Ingram (Ed.), *Information processing approaches to psychopathology and clinical psychology* (pp. 23-49). New York: Academic Press.

LAKATOS, I. (1970). Falsification and the methodology of science research programs. In I. Lakatos & A. Musgrave, *Criticism and the growth of knowledge* (pp. 91-195). Cambridge, England: Cambridge University Press.

LINDSAY, R. K., BUCHANAN, B. G., FEIGENBAUM, E. A., & LEDERBERG, J. (1980). *Applications of artificial intelligence for organic chemistry: The DENDRAL project*. New York: McGraw-Hill.

MCCARTHY, J. (1958). Programs with common sense. In M. L. Minsky (Ed.), *Proceedings of the symposium on the mechanization of thought processes* (pp. 403-409). Cambridge, MA: MIT Press.

MINSKY, M. (1967). *Computation: Finite and infinite machines*. Englewood Cliffs, NJ: Prentice-Hall.

NEWELL A. (1980). Physical symbol systems. *Cognitive Science*, **4**, 135-183.

NEWELL, A. (1984). Forward. In B. G. Buchanan & E. H. Shortliffe (Eds.), *Rule-based expert systems* (pp. xi-xvi). Reading, MA: Addison-Wesley.

NEWELL, A., & SIMON, H. A. (1956). The logic theory machine: A complex information processing system. *IRE Transactions on Information Processing*, **IT-2**, 61-79.

NEWELL, A., & SIMON, H. A. (1961). GPS, a program that simulates human thought. In H. Billing (Ed.), *Lernende Automaten* (pp. 109-124). Munich: R. Oldenberg.

NEWELL, A., & SIMON, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

NEWELL, A., & SIMON, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, **19**, 113-126.

NILSSON, N. J. (1980). *Principles of artificial intelligence*. Palo Alto, CA: Tioga Press.

PEARCE, J. M. S. (1984). *Dementia: A clinical approach*. Oxford, England: Blackwell.

POST, E. L. (1936). Finite combinatory processes—Formulation I. *Journal of Symbolic Logic*, **1**, 103-105.

SHAMOIAN, C. A. (Ed.) (1984). *Dementia in the elderly*. Washington, DC: American Psychiatric Press.

TURING, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (Series 2)*, **42**, 230-265.

VAN MELLE, W., SHORTLIFFE, E. H., & BUCHANAN, B. G. (1984). EMYCIN: A knowledge engineer's tool for constructing rule-based expert systems. In B. G. Buchanan & E. H. Shortliffe (Eds.), *Rule-based expert systems* (pp. 314-328). Reading, MA: Addison-Wesley.

WINOGRAD, T. (1982). *Language as a cognitive process*. Reading, MA: Addison-Wesley.