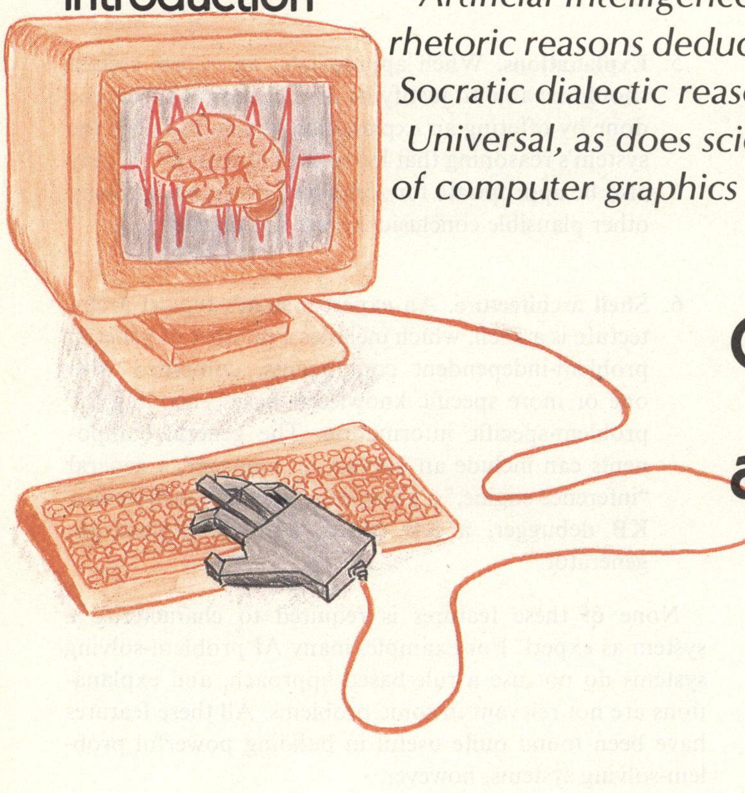
Guest Editors' Introduction

Backward and forward chaining—our nascent science of Artificial Intelligence—has ancient antecedents: Aristotelian rhetoric reasons deductively from the Universal to the Particular; Socratic dialectic reasons inductively from the Particular to the Universal, as does scientific method. The seedling confluence of computer graphics and Artificial Intelligence has deep roots.



Computer Graphics and Expert Systems

Norman I. Badler and Timothy W. Finin
University of Pennsylvania

As computer graphics evolves and expands into an essential component of many interactive systems, it is almost inevitable that the applications themselves become more complex and ambitious. In particular, the *interactive system* is rapidly becoming the *intelligent system:* Not only does the interface permit convenient access to the entities and operations of the application, but it also aids and augments the mere interaction in significant ways. For example, computer-aided design systems check topological constraints while an object is being constructed, robot simulators assist in avoiding collisions, and slide-making systems have automatic layout modes.

The implication of these aids is that the new generation of graphical systems will provide increasing intelligence in the human-computer interface. The study of the characteristics and capabilities necessary to effect communication between human and machine on this higher level is a central aspect of Artificial Intelligence (AI). One of our principal goals in presenting this special issue is to bring the approaching confluence of computer graphics and artificial intelligence into sharper focus. In addition, we hope to spur interest in common issues and open new pathways for future graphical systems.

The fields of artificial intelligence and computer graphics have interacted in a number of ways in the past. This interaction has been particularly strong in the areas of image processing and robotics. Image processing and graphics are naturally related because they share a common problem: How does the three-dimensional world appear when projected onto two dimensions. Another relation, one shared by robotics, is that graphics provides an important tool for presenting the data being processed by these systems as well as their internal states.

Other areas within AI also have a natural relationship with computer graphics. Natural language processing, for example, was once seen as offering a kind of interface that would be an alternative to a graphic one. The current trend, however, is to build sophisticated interfaces that combine both graphic and linguistic modalities. The relationship between graphics and natural language processing becomes more interesting when one looks at the underlying issue—communication. The study of natural language processing is really the study of communication between intelligent agents. In dealing with this larger problem of communication, researchers have developed a number of theories and techniques directly relevant to systems that communicate with people through graphics as well.

The papers presented in this special issue of *IEEE CG&A* involve the interaction of computer graphics and the relatively newer subfield of AI, expert systems. Each of the three systems described in this special issue is offered as a graphic interface to an expert system. Moreover, each system, although primarily designed as a graphic interface, makes good use of some of the new techniques that have come out of AI research. Thus, each can be characterized as an expert system.

Expert systems

An expert system could be defined as a computer program which solves problems in some particular domain; these problems normally require some specialized knowledge (e.g.,expertise) on the part of humans who successfully solve them. The operant terms in this definition are *solves problems, particular domain*, and *specialized knowledge.* This definition would exclude programs that are not characteristically problem solvers (e.g., a text editor), attempt to be very general (e.g., a theorem prover), solve problems that are considered straightforward implementations of well-defined theories (e.g., computing fast Fourier transforms), or are just mundane (e.g., simple record-keeping). Typical domains for expert systems are medical diagnosis, signal interpretation, fault diagnosis, and computer configuration.

Unfortunately this definition is still much too broad: It encompasses most of what used to be called "application programs." To capture what is new and valuable in recent work on expert systems, it is useful to enumerate some of the major features found in "good" examples of expert systems. In addition to the partial definition offered above, a good example of an expert system would be likely to have the following characteristics:

1. **Separate knowledge base.** The special domain knowledge that the expert system uses is explicitly represented in a module (the knowledge base or KB) which is separate from the components that use it.
2. **Multiple use of knowledge.** Since the KB is explicitly represented as a separate module, it can be used in several different ways: to make decisions, to construct explanations, to construct tutorials, etc. This requires that the knowledge be represented in a more general way so that it does not favor one use at the expense of another. The goal is to express the knowledge in a general way that allows it to be reasoned about as well as reasoned with.
3. **Special knowledge representation languages.** The knowledge in an expert system is usually encoded in a special representation language. Most current AI systems use representation languages that rely on one or more of three general techniques: rules, frames, and logical relations.
4. **In the knowledge lies the power.** This often-quoted slogan underscores the fact that the real intelligence in the expert system lies in the domain knowledge represented in its KB, not in any of the more general components. As a corollary, any search strategies for problem solutions should be expressed in the general KB rather than in the code that uses the KB.
5. **Explanations.** When appropriate, an expert system should be able to justify its conclusions. This can be done by offering an explanation which describes the system's reasoning that led to the conclusions. It may also be appropriate for some systems to explain why other plausible conclusions were not reached.
6. **Shell architecture.** An expert system's typical architecture is a shell, which includes a number of general, problem-independent components, combined with one or more specific knowledge bases encoding the problem-specific information. The general components can include an interaction manager, a general "inference engine," a knowledge-acquisition system, a KB debugger, a KB editor, and an explanation generator.

None of these features is required to characterize a system as expert. For example, many AI problem-solving systems do not use a rule-based approach, and explanations are not relevant in some problems. All these features have been found quite useful in building powerful problem-solving systems, however.

The origin of expert systems

Our current notion of an expert system as a natural class of computer programs was first articulated in the early 1970's by Edward Feigenbaum and his colleagues at the Heuristic Programming Project (HPP) of Stanford University. Their work on problem solving in AI was a methodological departure from preceding research, which had focused on the development of general problem-solving strategies that could be used in conjunction with detailed descriptions of a particular problem to produce a solution. Feigenbaum and his colleagues at HPP accepted the legitimacy of representing and using as much problem-specific knowledge as possible. Thus they de-emphasized the attempt to discover and use general principals of intelligence and accepted the utility of building in problem-specific expertise.

The first system this group produced was called *Heuristic Dendral.* This system attempted to identify likely chemical structures for unknown compounds based on their chemical formulae and data from a mass spectrogram. Besides acknowledging the need for domain-specific expertise, Dendral pioneered the representation of this knowledge in the form of rules—a technique that has turned out to be important for many subsequent expert systems.

This technique was further developed by the HPP group in the expert system MYCIN, which assisted physicians in choosing appropriate therapies for patients suspected of having bacterial infections. Besides encoding its knowledge in the form of rules, the MYCIN development made important advances in the areas of reasoning with uncertain facts and rules, generating explanations for conclusions, managing a consultation dialogue, and providing environments for knowledge acquisition.

Rule-based systems

In simple terms, a rule-based system has two major components: a knowledge base and an inference engine. The knowledge base contains a set of facts and a set of rules, which together represent the system's general domain knowledge and specific knowledge about the current problem. A fact is generally considered to be an atomic proposition which is true in the world, and a rule is considered a conditional with an antecedent (if) part and a consequent (then) part. Thus a rule is knowledge of the form:

IF/antecedent—THEN/consequent

Depending on the nature of the antecedent and consequent parts, a rule can be one of several different varieties. If both parts are viewed as logical propositions, then the rule can be viewed as a rule of inference. If the antecedent is interpreted as a partial description of some state of the factual knowledge base and the consequent can be any executable expression, then the rule is similar to the notion of a production rule as used by cognitive scientists. Many rule-based systems have a need to deal with uncertain data and rules which do not always hold. A common variation, then, is to associate with each fact or rule a *degree of certainty* which represents the system's confidence that the fact or rule is true.

The inference engine is a kind of rule interpreter which takes the set of facts and rules and computes additional facts or rules that hold. An inference engine can use rules in one of two important ways. Forward chaining reasons from the initial set of facts to derive additional facts which must hold. The inference engine identifies rules whose antecedent parts are satisfied by the facts currently in the KB. Backward chaining reasons from a given goal backward to a set of facts which, if in the current KB, would support the goal. Given a goal to satisfy, the inference engine looks for an appropriate fact or rule in the KB. If a matching fact is found, then the goal has been satisfied. If a rule is found in which the consequent matches the goal, then the engine attempts to satisfy the rule's antecedent recursively by setting it up as a subgoal.

The articles

In the first article of this special issue, Steve Feiner describes a system (APEX) that generates pictures portraying the performance by a problem solver of physical actions in a three-dimensional world. As a graphic system, APEX relates to the notion of an AI expert system in two ways: First, APEX was designed to generate pictures from the output of an expert system that plans physical actions. Second, APEX is a kind of expert system itself. It takes a general description of a set of physical objects, an action to be performed on them, and a simple model of a human's knowledge of the scene and determines what would be a good graphic description of the planned action for that person. To do this, it must decide what objects should be included in the picture, how much detail must be used, how the action can be indicated, and a host of other factors.

The second article describes a system called SAGE, developed by Eric Clemons and Arnold Greenfield. Their system is designed as a general framework for building graphic interfaces for a wide class of sensitivity analysis models—models that certainly fall within the general definition of an expert system. The authors make the point that sensitivity analysis systems require their users to make changes in the model and quickly evaluate the results. The graphical presentation of the effects of a model change aid the quick evaluation.

The final article describes GUIDON-WATCH, a graphic interface for browsing and viewing a class of expert systems based on NEOMYCIN. The authors have included a number of techniques for displaying the various components of a large, complex expert system. GUIDON-WATCH exploits trees, tables, icons, animation, and other graphic formats in a window-based environment. The result is an interface that helps both the end-user and the system designer understand, debug, and modify a large expert system.

For more information

Some sources for more information on artificial intelligence, expert systems, and knowledge representation follow:

"Special Section on Architectures for Knowledge-Based Systems," *Communications of the ACM*, Vol. 28, No. 9, Sept. 1985.

This issue contains three tutorial articles on the three dominant paradigms for building expert systems—frames, rules, and logic.

Brachman, R., and H. Levesque, eds., *Readings In Knowledge Representation*, Morgan Kaufman Publishers, Los Altos, Calif., 1985.

This book contains about 30 seminal articles on various issues concerning the representation of knowledge. These articles are representative of most of the important ideas developed over the last 15 years.

Charniak, E., and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, Mass., 1985.

This is a good general introduction to AI, emphasizing the use of logic and deduction.

Hayes-Roth, F., D. Waterman, and D. Lenat, eds., *Building Expert Systems*, Addison-Wesley, Reading, Mass., 1983.

This book contains articles that address both theoretical and practical issues. An interesting feature of the book is that it presents a typical expert system problem and shows how a number of different expert system building tools could be employed to solve it.

Special Issue on Knowledge Representation, *Computer*, Vol. 16, No. 10, Oct. 1983.

This special issue of *Computer* gives a good picture of the current state of the art and current research areas in knowledge representation.

Kowalski, R., *Logic for Problem Solving*, Elsevier, New York, 1979.

This classic book presents the use of logic as a language for representing knowledge and solving problems.

Shortliffe, E., and B. Buchanan, *Rule-Based Expert Systems*, Addison-Wesley, Reading, Mass., 1984.

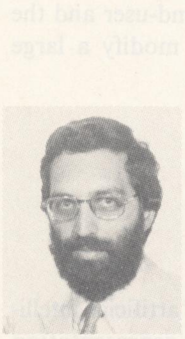
This edited collection of articles, based on research done at the Heuristic Programming Project at Stanford, discusses the important expert system MYCIN and its descendants, as well as general issues.

Waterman, D., *A Guide to Expert Systems*, Addison-Wesley, Reading, Mass., 1985.

This book attempts to introduce and explain expert systems to readers who do not have a background in AI. It contains a comprehensive catalog with descriptions of over 200 expert systems.

Weiss, S., and C. Kulikowski, *A Practical Guide to Designing Expert Systems*, Rowman and Allanheld, Totowa, NJ, 1984.

This book is based on the experience of Weiss, Kulikowski, and their colleagues at Rutgers University, where they built expert systems using the system EXPERT.



Norman I. Badler is Associate Professor of computer and information science at the University of Pennsylvania, and has been on that faculty since 1974. Badler is also a coprincipal investigator of research projects for the National Science Foundation and has a US Army Research Office artificial intelligence grant.

Badler was named associate editor-in-chief of *IEEE CG&A* earlier this year, and has been an active participant in ACM Siggraph since 1975. He was conference tutorial chair from 1976 to 1979, and was elected vice-chair in 1979 and again in 1981. He is the coauthor of more than 35 technical papers, and operates a computer graphics research facility with full-time staff and about 25 student participants.

Badler received his BA in creative studies mathematics from the University of California-Santa Barbara, and his MS in mathematics and PhD in computer science from the University of Toronto.

Timothy W. Finin is an Assistant Professor of computer and information science at the University of Pennsylvania, where he has been a faculty member since 1980. His research interests include computational linguistics, knowledge representation, and expert systems. In 1983 he was awarded an IBM faculty development award.

Finin received the BS in electrical engineering from the Massachusetts Institute of Technology, and the MS and PhD in computer science from the University of Illinois. He did computer vision research at MIT's Artificial Intelligence Laboratory from 1971 to 1974, and has worked in natural language processing with the Coordinated Science Laboratory.

The authors' address is University of Pennsylvania, Computer and Information Science, School of Engineering and Applied Science, Moore School D2, Philadelphia, PA 19104.