Expert (Knowledge-Based) Systems

K

Petr Berka
University of Economics, Prague, Czech Republic & University of Finance and Administration, Prague, Czech Republic

INTRODUCTION

Knowledge representation and knowledge engineering are central to AI (artificial intelligence) research. Many of the problems machines are expected to solve will require extensive knowledge about the world. This fact was realized in the mid-1970s and implemented in the so-called expert systems, “computer programs that emulate the decision-making ability of a human expert” (Jackson, 1990, p. 2). This article covers the following topics:

- Basic notions from the area of expert systems
- Types of expert systems (diagnostic, generative) and examples of their applications
- Architecture of an expert system (knowledge base, inference mechanism, uncertainty processing)
- Building expert systems (life cycle, KADS methodology)
- Knowledge engineering

BACKGROUND

Expert systems represented the prominent research area within AI in the 1970s. In these times the search for a general problem-solving algorithm (using the formalism of the state space) has encountered its limitations in the domains that required specialized domain knowledge.

An expert system is:

An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution (Feigenbaum, 1979).

Expert system is a branch of AI that makes extensive use of specialized knowledge to solve problems at the level of a human expert (Girratano & Riley, 1993, p. 2).

An expert system is a knowledge-based program that provides ‘expert quality’ solutions to problems in a specific domain (Luger & Stubblefield, 1989, p. 291).

The power of an expert system (ES) is derived from presence of a knowledge base filled with expert knowledge, mostly in symbolic form. In addition, there is a generic problem-solving mechanism used as an inference engine. Some other more-or-less typical features of expert systems are uncertainty processing, dialogue mode of the consultation, and explanation abilities.

EXPERT SYSTEMS

Expert systems should substitute human experts in the decision-making process. An expert system can play the following roles in this process:

- Expert
- Colleague
- Assistant

The expectation that expert systems will play the role of an expert was too optimistic and unrealistic. The main reason for the fact that expert systems cannot always guarantee decision-making abilities of an expert is that a lot of knowledge of a real expert has a tacit form and thus cannot be transferred into the system. Another reason is that the users considering themselves to be experts were reluctant to follow the recommendations of the system if they disagree. So expert systems have to play either the role of a colleague or the role of an assistant. In the first case, the system has the same knowledge as its user, but as it never

DOI: 10.4018/978-1-4666-5888-2.ch446

works under stress, never forgets, and can more quickly enumerate all possible solutions, the users check their decisions with the system, if necessary. In the latter case, the user is in fact an expert when compared with the knowledge of the system, but the system is useful to solve relatively simple, routine problems that usually prevail in various application areas.

Types of Expert Systems

We can distinguish between two basic types of expert systems: diagnostic and generative. The main difference between these types is that in *diagnostic systems* (or applications) the knowledge base contains the list of outcomes (diagnoses, recommendations) the system chooses from, whereas in *generative systems* these outcomes are created during the consultation. So, a typical diagnostic application is medical diagnosis when the system assigns diagnoses to observed symptoms (the first expert system, MYCIN, was designed to identify bacteria causing severe infections and to recommend antibiotics; the system ONCOCYN was used to plan the chemotherapy for patients with cancer; and the system INTERNIST/CADUCEUS was used to diagnose internal medicine diseases), or interpretation of signals and measurements (the expert system PROSPECTOR helped to discover a mineral deposit worth 100 million dollars). A typical generative application is design (the system R1/XCON has been used by DEC Inc. to configure their mainframe computers according to customer requests), or planning (the system MOLGEN was used to plan experiments in the area of molecular biology).

EXPERT SYSTEM ARCHITECTURE

An expert system consists of two main parts: knowledge base and inference mechanism. The inference mechanism is a domain-independent algorithm used for reasoning, while the knowledge base contains domain-specific knowledge acquired from the experts. As these two parts are separated, it is possible to create so-called empty expert systems (also called expert system shells) that can be "filled in" with different knowledge bases and thus be used in various domains. First example of an empty system is EMYCIN created from MYCIN by removing the domain-specific knowledge. The next part is the working memory that contains current data (questions of the system answered so far, partial results). If run in dialogue mode, the expert system also contains the communication and explanation modules. In this mode, the system asks questions about truth degrees of some propositions (in the simplest way the answers can be yes or no) in a way similar to that of a human expert (physician, financial expert, etc.). In this setting the user can also ask for some explanations concerning the asked question or the inferred results.

Knowledge Representation

Knowledge base contains domain knowledge. Possible representation formalisms include

- **Predicate Logic:** This is a traditional formalism used in mathematics for centuries to define new terms.
- **Semantic Nets:** This formalism was originally proposed to capture the semantics of natural language sentences and later extended to express knowledge (in general) in the form of a graph where nodes represent objects and edges represent relations.
- **Frames:** Frames have been designed in the mid-1970s to represent stereotypical situations. The basic features of frames, i.e., inheritance in a hierarchy of concepts, encapsulation and polymorphisms have been reused in object oriented programming languages that thus provide a tool suitable for representing knowledge.
- **Rules:** Rules are inspired by IF-THEN statements from various programming languages or by implications from propositional logic. The nature of a rule can be procedural (as used in generative systems), i.e.,

  IF situation THEN action

  or declarative (as used in diagnostic systems), i.e.,

  IF condition THEN conclusion

  The situation or condition is a combination (usually conjunction) of statements, the action is a list of actions that can be performed if the respective situation occurs, and the conclusion