11

APPROXIMATE REASONING

11.1 FUZZY EXPERT SYSTEMS: AN OVERVIEW

An expert system, as the name suggests, is a computer-based system that emulates the reasoning process of a human expert within a specific domain of knowledge. Expert systems are primarily built for the purpose of making the experience, understanding, and problem-solving capabilities of the expert in a particular subject area available to the nonexpert in this area. In addition, they may be designed for various specific activities, such as consulting, diagnosis, learning, decision support, design, planning, or research.

A typical architecture of an expert system is depicted by the block diagram in Fig. 11.1. Let us describe the role of each of the units shown in the diagram. Our focus is, of course, on fuzzy expert systems.

The kernel of any expert system consists of a knowledge base (also called a long-term memory), a database (also called a short-term memory or a blackboard interface), and an inference engine. These three units, together with some interface for communicating with the user, form the minimal configuration that may still be called an expert system.

The *knowledge base* contains general knowledge pertaining to the problem domain. In fuzzy expert systems, the knowledge is usually represented by a set of *fuzzy production rules*, which connect antecedents with consequences, premises with conclusions, or conditions with actions. They most commonly have the form *"If A, then B,"* where A and B are fuzzy sets.

The purpose of the *database* is to store data for each specific task of the expert system. The data may be obtained through a dialog between the expert system and the user. Typically, such data are parameters of the problem or other relevant facts. Other data may be obtained by the inference of the expert system.

The *inference engine* of a fuzzy expert system operates on a series of production rules and makes fuzzy inferences. There exist two approaches to evaluating relevant production rules. The first is data-driven and is exemplified by the generalized modus ponens. In this case, available data are supplied to the expert system, which then uses them to evaluate relevant production rules and draw all possible conclusions. An alternative method of evaluation is goal-driven; it is exemplified by the generalized modus tollens form of logical

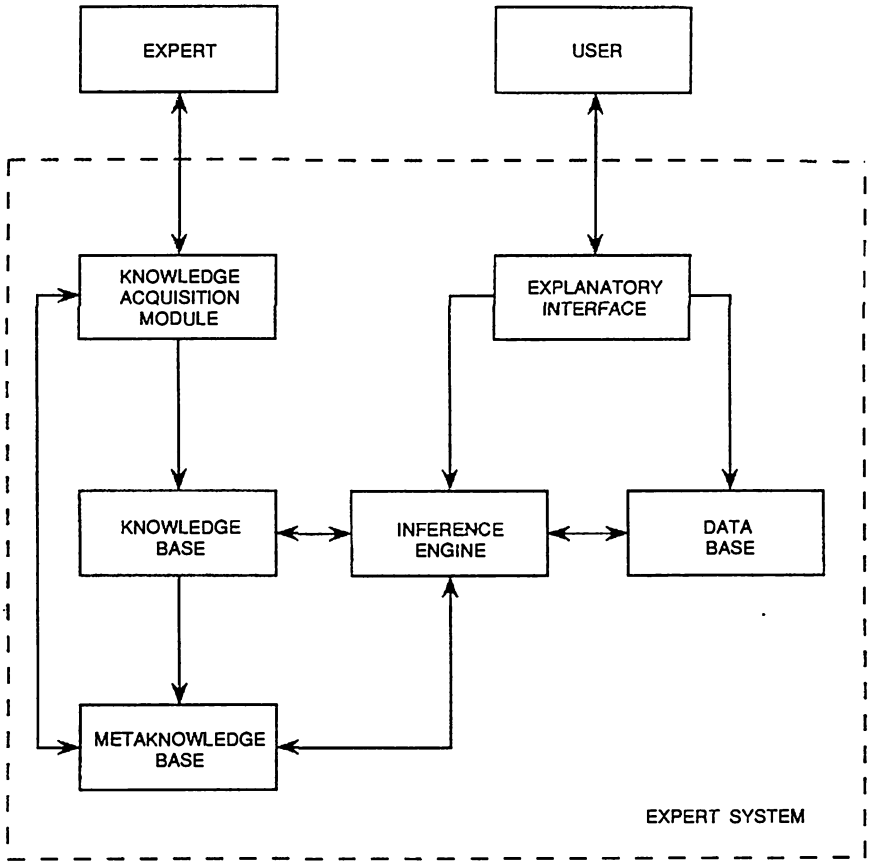

Figure 11.1 Architecture of an expert system.

inference. Here, the expert system searches for data specified in the *IF* clauses of production rules that will lead to the objective; these data are found either in the knowledge base, in the *THEN* clauses of other production rules, or by querying the user. Since the data-driven method proceeds from *IF* clauses to *THEN* clauses in the chain through the production rules, it is commonly called *forward chaining*. Similarly, since the goal-driven method proceeds backward, from the *THEN* clauses (objectives) to the *IF* clauses, in its search for the required data, it is commonly referred to as *backward chaining*. Backward chaining has the advantage of speed, since only the rules leading to the objective need to be evaluated. Moreover, if certain data are difficult to obtain, and these are only potentially necessary, then the backward-chaining method is clearly superior.

The inference engine may also use knowledge regarding the fuzzy production rules in the knowledge base. This type of knowledge, whose appropriate name is *metaknowledge*, is located in the unit called a *metaknowledge base*. This unit contains rules about the use

of production rules in the knowledge base. These rules, or rather *metarules*, prescribe, for example, stopping criteria, require precedences in applying certain production rules under various conditions or whether a needed fact should be inferred or requested from the user. The primary purpose of the metaknowledge base is to simplify computation by pruning unnecessary paths in the search space.

The *explanatory interface* facilitates communication between the user and the expert system. It enables the user to determine how the expert system obtained various intermediate or final conclusions, or why specific information is being requested from the user. This capability is crucial for building user confidence in the expert system. It is also very important for the identification of errors, omissions, inconsistencies, and so on, during the debugging of the knowledge base or inference engine.

The *knowledge acquisition module*, which is included only in some expert systems, makes it possible to update the knowledge base or metaknowledge base through interaction with relevant human experts. In general, this unit must implement suitable algorithms for machine learning, such as algorithms conceptualized in terms of artificial neural networks (Appendix A) or genetic algorithms (Appendix B), by which fuzzy productions can be learned from examples obtained from human experts. This capability allows the expert system to expand or modify its knowledge base or metaknowledge base through feedback during operation.

When the knowledge domain is removed from an expert system, the remaining structure is usually referred to as an *expert system shell*. The applicability of an expert system shell is not necessarily restricted to one particular knowledge domain. An inference engine embedded in an appropriate expert system shell is thus, in principle, reusable for different domains of knowledge and, thus, for different expert systems.

The purpose of this chapter is to cover fundamentals of reasoning based on fuzzy production rules, which is usually referred to as *approximate reasoning*. This material is essential for the design of inference engines for fuzzy expert systems. The actual design of fuzzy expert systems or even inference engines for approximate reasoning is beyond the scope of this book. However, we guide the reader through the literature on this topic in Note 11.1.

11.2 FUZZY IMPLICATIONS

The logic operation of implication is as essential for approximate reasoning as it is for reasoning within classical two-valued logic. In general, a *fuzzy implication*, $\mathcal{J}$, is a function of the form

$$\mathcal{J} : [0, 1] \times [0, 1] \rightarrow [0, 1],$$

which for any possible truth values a, b of given fuzzy propositions p, q, respectively, defines the truth value, $\mathcal{J}(a, b)$, of the conditional proposition "if p, then q." This function should be an extension of the classical implication, $p \Rightarrow q$, from the restricted domain $\{0, 1\}$ to the full domain $[0, 1]$ of truth values in fuzzy logic.

In classical logic, where $a, b \in \{0, 1\}$, $\mathcal{J}$ can be defined in several distinct forms. While these forms are equivalent in classical logic, their extensions to fuzzy logic are not equivalent and result in distinct classes of fuzzy implications. This fact makes the concept of fuzzy implication somewhat complicated. The purpose of this section is to discuss the