COUNSELOR EXPERT SYSTEM

Debretsion G.Michael and Devarajan G.
Department of Electrical and Computer Engineering
Addis Ababa University

ABSTRACT

An expert system plays an important role on alleviating primarily shortage of experts in a specific area of interest. With the help of an expert system, personnel with little expertise can solve problems that require expert knowledge. In this paper all major aspects of an expert system development have been presented.

The different types and components of an expert system is discussed in this paper. Sequence of steps to be followed in developing an expert system has also been given due attention. Knowledge acquisition, which is the most crucial phase of expert system development has been addressed extensively. Different modes of knowledge representation and the inference engine used for reaching a goal (conclusion) have also been discussed in detail.

INTRODUCTION

An expert system can be broadly defined as a computer system consisting of both (hardware and software) that simulates human experts in a given area of specialization. As such an expert system should be able to process and memorize information, learn and reason in both deterministic and uncertain situations, communicate with human and/ or other expert systems, make appropriate decisions, and explain why these decisions have been made. One can also think of an expert system as a consultant that can provide help to (or in some cases completely substitute) the human beings with a high degree of reliability.

There are several reasons for using expert systems:

- With the help of an expert system, personnel with little expertise can solve problems that require expert knowledge.
- The knowledge of several human experts can be combined together which gives rise to a more reliable expert system, a system that is based on the collective. wisdom of several experts, rather than on the experience of a single expert.
- Expert systems can answer questions and solve problems much faster than the human expert.
- Expert systems can provide both fast and reliable answers in situations where the human experts cannot.
- Expert systems can be used to perform monotonous operations and others that are boring or uncomfortable to humans. Indeed, expert systems may be the only viable option in a situation where the task to be performed may jeopardize a human life.
- Substantial savings in terms of cost and time can be achieved from using expert systems.

Expert systems have already been constructed and are being sold and/or implemented in such diverse areas as follows:

- Stock market advisors
- Financial planning
- Tax preparation and planning
- Granting of loans and determination of credit limits
- Diagnosis and treatment of various diseases
- Determination of the chemical properties of unknown compounds
- Scheduling and control of the automated factory
- Diagnosis and maintenance of complex machinery (e.g., locomotives, aircraft, spacecraft, and ships)
- Assignment of planes to airport gates and the scheduling of flights
- Layout and design of printer-circuit boards
- Facility location and layout
- Automation of the auditing procedure for foreign exchange transactions

While the above list is impressive, it is by no means complete nor does it indicate the vast potential for additional implementations of expert systems [2].

In the following consecutive sections of this paper different issues of expert system has been covered. Section II of the paper deals with the types, components and design steps of an expert system. Section III discusses basic mathematics of expert system and addresses the main elements to be covered. How knowledge can be extracted from experts in developing expert system is dealt with in section IV. Here an attempt has been made to explore the possibilities for extracting accumulated experiences used for constructing the knowledge based expert system [1]. In section V the means of representing knowledge has been discussed. The inference mechanism used is dealt with briefly in section VI. Finally, the conclusion drawn from this work is presented in section VII.

AN EXPERT SYSTEM

(a) Types of Expert System

The problems that expert systems can deal with can be broadly classified into two types: mainly deterministic and stochastic problems. Consequently, expert systems can be classified into two main types according to the nature of the problems they are designed to solve, i.e., and deterministic and stochastic expert systems. Deterministic problems can be formulated using a set of rules that relates several well-defined objects. Expert systems that deal with deterministic problems are known as rule based expert systems because they draw their conclusion based on a set of rules using a logical reasoning mechanism.

In stochastic situations it is necessary to introduce some means for dealing with uncertainty. Several uncertainty measures have been proposed during the last decades. Some of these measures include certainty factor, used in expert system shells and fuzzy logic. One intuitive measure of certainty is probability, where a joint probability distribution of a set of variables is used to describe the relationships among the variables, and conclusions are drawn using certain well-known probability formulas.
Expert systems that use probability as a measure of uncertainty are known as probabilistic expert systems, and the reasoning strategy they use is known as probabilistic reasoning or probabilistic inference.

(b) Components of an Expert System

The building blocks of expert systems are shown schematically in Fig. 1. They are:

- **The Human Component:** The human experts provide the knowledge based in subject matter area and the knowledge engineers translate this knowledge into a language that the expert system can understand.
- **The Knowledge Base:** Knowledge can be either abstract or concrete. Abstract knowledge refers to statements of general validity such as rules, probability distributions etc. Concrete knowledge refers to information related to a particular application. For example, in medical diagnosis, the symptoms and diseases and relationship among them form the abstract knowledge, whereas particular symptoms of a given patient form the concrete knowledge. The abstract knowledge is stored in the knowledge base, and the concrete knowledge is stored in the working memory. All procedures of the different systems and subsystems that are of a transient character are also stored in the working memory.
- **Knowledge Acquisition Subsystem** The knowledge acquisition subsystem controls the flow of new knowledge from the human experts to the knowledge base.
- **Coherence Control:** The Coherence control subsystem has appeared in expert systems only recently. This subsystem controls the consistency of the knowledge base and prevents any incoherence knowledge from reaching the knowledge base.
- **The Inference Engine:** The inference engine is the heart of every expert system. The main purpose of this component is to draw conclusions by applying the abstract knowledge to the concrete knowledge.
- **The Information Acquisition:** If the initial knowledge is very limited and conclusions cannot be reached, the inference engine utilizes the information acquisition subsystem in order to obtain the required knowledge and resume the inference process until conclusions can be reached.
- **User Interface:** The user interface subsystem is the liaison between the expert system and the user. Thus, in order to communicate effectively with an expert system, it must incorporate efficient mechanisms to display and retrieve information in an easy way. When the inference engine due to lack of information can reach no

conclusion, the user interface provides a vehicle for obtaining the much-needed detailed information from the user.

- **The Action Execution Subsystem:** The action execution subsystem is the component that enables the expert system to take actions. These actions are based on the conclusions drawn by the inference engine.
- **The Explanation Subsystem:** An explanation subsystem is needed to explain the process followed by the inference engine or by the action exeeution subsystem.
- **The learning Subsystem:** One of the main features of an expert system is the ability to learn from the past experience. Another feature of expert systems is their ability to gain experience, based on available data. These data can be collected by both experts and non-experts and can be used by the knowledge acquisition subsystem and by the learning subsystem.

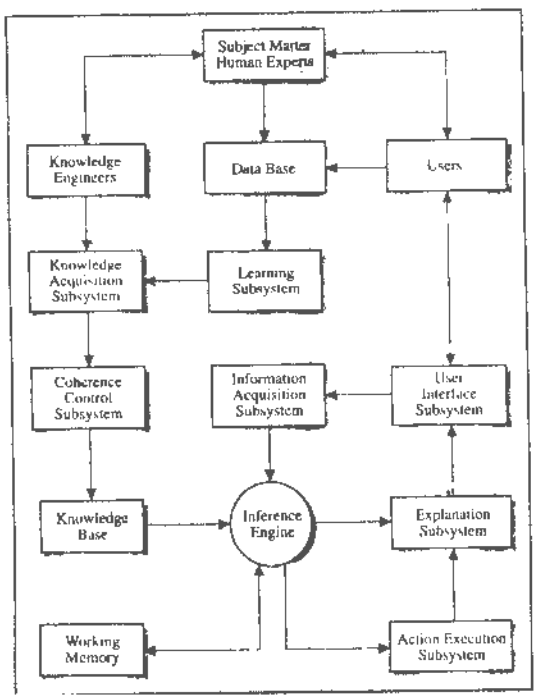

Figure 1 Components of an expert system

(c) Development of an Expert System

The main sequence of steps that should be followed for the design and implementation of an expert system are shown in Fig. 2.

1. **Statement of the problem.** The first step in any project is that usually the definition of the problem is to be stated clearly. This is the basis for building an expert system. Here we can impose constraints, limitations of the system.

2. **Finding human experts to a specific problem**. As we know the expert system is built only in the domain/field where limited number of experts are available. The design engineer has to identify the experts in that field. The domain expert identified should be willing to share, divulge his knowledge for the future growth of the expert system and also for further advancement in the domain. Some experts may not be willing to divulge their knowledge due to professional rivalry or may not be willing to spend time on this, as the expert has nothing much to gain personally out of this work. The human expert knowledge is one of the main sources of database for building expert system. The design engineer plays a key role to extract the knowledge from the expert and put it in the form of database.

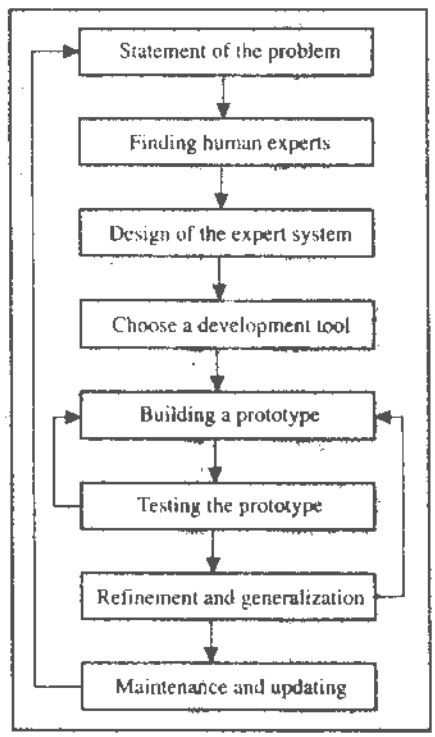

Figure 2 Expert system development steps

3. **Design of the expert system**. This step includes designing the structures for knowledge storage, the inference engine, the explanation subsystem, the user interface etc.

4. **Choosing the development tool, shell, or programming language.** A decision has to be made between developing a specially designed expert system, a shell, a tool, or a programming language. If a tool or shell satisfying all design requirements exists, it should be used not only for financial reasons but also due to reliability implications.

5. **Developing and testing of a prototype.** If the prototype does not provide the desired result/outcome then the previous steps (with the appropriate modifications) have to be repeated until a satisfactory prototype is obtained.

6. **Refinement and generalization.** In this step the rules or knowledge which are ambiguous are modified to be more meaningful to the domain.

7. **Maintenance and updating.** In this step user's complaints and problems must be taken into consideration, as well as correction of bugs and errors, updating of the product with new advances, etc.

FUZZY MATHEMATICS

(a) Fuzzy Logic

In conventional logic, a statement is either true or false, with nothing in between. Fuzzy logic offers a better way of representing reality. In fuzzy logic, a statement is true to various degrees, ranging from completely true (1) through half-truth to completely false (0).

The basic idea of multi-valued logic has been explored to some extent by a number of mathematicians in this century, but Prof. Lotfi Zadeh of the University of California in Berkeley [3] made the real breakthrough. In 1965 he published a paper on the theory of fuzzy sets; that paper has given rise to hundreds of papers on fuzzy mathematics and fuzzy systems theory.

It is possible to reason in terms of words, such as *small, medium, fast, slow* and so on, relative to a particular object rather than in terms of numbers; ambiguities and contradictions can be easily handled; and uncertainties pose no problems.

Uncertainties are handled by assigning confidence factor (CF). Confidence factor is assigning a number value between 0 and 1 to an event. Assigning **1** is meant that one is absolutely certain that the event will happen, while assigning **0** meant that the event has absolutely no chance to happen. Numbers between 0 and 1 indicate various levels of uncertainty concerning the occurrence of the specified event.

Consider now the truth of the combination of two statements, A AND B. A and B are both assertions, In conventional logic, of course both A and B must be either true or false. The statement (A AND B) is true only if both A and B is individually true; otherwise, the statement (A AND B) is false. Fuzzy logic gives a remarkably simple answer to this problem: the truth of (A AND B) together is the minimum of the truth-value of A and the truth-value of B. Using the expression of fuzzy logic, i, e, the confidence level (CL), which is utilized for taking decisions, we have:

$$CL \quad A \wedge B - \min (A \wedge B)$$

To demonstrate, consider the following production rule:

Rule 1: If disk drive is noisy (A) (0.8) and disk sector formatting results in bad (B) (0.3) then disk drive status defective is (0.9)
Then
C = A ^ B – min (A ^ B)
= min (0.8 ^ 0.3)
0.3
Thus, our confidence in rule 1's conclusion is simply, 0.9 x 0.3 0.27.

Similarly, consider the statement (A OR B), which in conventional logic is true if statement A or statement B or both are true, and is only false if both A and B are false. In fuzzy logic, the truth of (A OR B) is simply the maximum of the truth-value of A and the truth-value of B. Using confidence level,

$$CL - A \vee B - \max (A \vee B)$$

Changing the rule 1 by or-ing:
C – A V B =max (A V B)
= max (0.3 V 0.8)
= 0.8
Now the confidence in the same rule becomes, 0.9 x 0.8 = 0.72.

These simple but new rules of logic have great impact on patterns of reasoning. It is with patterns of reasoning that we are most concerned in constructing expert systems; it is patterns of real-life productive human reasoning that we are trying to emulate. Fuzzy logic gives us the theoretical tools to do this.

(b) Fuzzy Sets

A set is simply a collection of items (objects) and each item is called an element of the set. The number of objects or items in a set is finite. Any set of objects chosen from a set is called subset. Sometimes the object may not belong to the set. Similar to the idea in logic those statements are either true or false. In 1965, Lotfi Zadeh proposed the idea of a *fuzzy set*; a fuzzy set is one to which objects can belong to different degrees, called *grades of membership*. This simple idea of different grades of membership in a fuzzy set is extremely helpful in constructing expert systems. We will be concerned with fuzzy sets of descriptive words, where the grade of membership represents our confidence that the descriptor is true of whatever we are considering.

(c) Fuzzy Numbers

The final major member of fuzzy systems arsenal is the *fuzzy number*. A fuzzy number lies between 0 and 1, whose precise value is somewhat uncertain. A very convenient way to describe fuzzy numbers is to use modifying words. For example, a *fuzzy two* could be completely specified by "roughly 2". Other modifying words available are "nearly", "about" and "crudely", with progressively larger uncertainties. These words are called *hedges* [3] in fuzzy math circles. With fuzzy numbers, we can make approximate comparisons.

KNOWLEDGE ACQUISITION

Knowledge acquisition is that phase of expert systems development dedicated to the identification of the rules and facts that comprise the knowledge base. In some cases, such acquisition may be accomplished through interviews with human experts. In others, human experts either do not exist or are unavailable. In this latter instance, one may either attempt to be one's own expert or utilize, if possible, historical data to construct a set of production rules. Yet another alternative is that we might consider training the domain expert or at least recognize the existence of problems that might be approached by expert systems.

The knowledge acquisition phase of development is also one that can be extremely frustrating as well as time consuming. Here, we are often dealing directly and intimately with domain experts. While dealing with people in general can be difficult, interfacing with domain experts can be many times as frustrating.

(a) Domain Expert

Domain expert is an individual who is believed to have significant expertise gathered over past many years of experience within the domain in question. After working with the domain expert for a few sessions, it should be possible to develop a simple, prototype expert system. Once a prototype is available, we may use it to extract additional knowledge from the expert. That is, once the expert is introduced to the prototype, a fuller appreciation of the concept and purpose of an expert system is obtained. Further, the prototype provides a tangible basis for evaluation. In particular, the expert can note inconsistencies, limitations, and deficiencies of the prototype, which in turn may be used to refine and enhance the rule base.

As more and more heuristic rules are entered into the knowledge base, and as more demonstration examples are encountered, the *intelligence* of the prototype will grow-first slowly, then faster and faster until a reliable decision analysis assistant has evolved.

(b) Knowledge Engineer

The knowledge engineer is the artificial intelligence (AI) language and representation expert. He/she should be able to identify a suitable expert system shell (and other tools) for the project, extract the knowledge from the expert, and implement the knowledge in a correct and efficient knowledge base.

Knowledge engineer has to first go through all the literature/manual available and gather all sort of information (deterministic, in deterministic nature of the knowledge) so that he can dialogue effectively, more fruitfully without irritating/wasting domain expert's precious time. From the dialogue carried with the expert, the knowledge engineer will extract the required knowledge. Preliminaries and facts related to the domain will be obtained from the dialogue. The knowledge engineer will assign CF based on the expert's experience and finally evaluate the expert system before sending it for field evaluation.

Extracting expert's knowledge is the most difficult part of expert system-building. In certain cases the expert will be able to clearly explain or define the symptoms/attributes. However in most cases the expert will use his own rule of thumb with some symptoms. Here the knowledge engineer has to probe in depth to extract more and more associated symptoms/attributes to form a better rule so that the expert system will be foolproof.

(c) Knowledge Acquisition via Rule Induction

An alternative to the acquisition of knowledge through the interface with a human (i.e., an expert or a knowledge engineer assuming the role of the expert) is to convert an existing (*and appropriate*) database into a set of production rules. The appropriate database, in turn, must consist of data that encompass examples pertaining to the type of problem under consideration-*where the examples selected should represent desirable outcomes*. More specifically, one needs examples of *good* decision making. In some cases, this approach may provide adequate results while, in other, it may at least lead to the development of a credible prototype system.

KNOWLEDGE REPRESENTATION

The field of knowledge representation is concerned with the mechanisms for representing and manipulating information or knowledge gathered from texts, conventions, and heuristics and from experts. Knowledge representation is crucial. Solving problems in a particular domain generally requires knowledge of the objects in the domain and knowledge of how to reason in that domain - both these types of knowledge must be intelligently represented.

Knowledge must be represented efficiently, and in a meaningful way. Efficiency is important, as it would be impossible to explicitly represent every fact that one might ever need or come across. There are just so many potentially useful facts, most of which one would never even think of. One have to be able to infer new facts from the existing knowledge, as and when needed, and capture general abstractions, which represent general features of sets of objects in the world.

Knowledge must be meaningfully represented so that we know how it relates back to the real world. A knowledge representation scheme provides a mapping from features of the world to a formal language. When we manipulate that formal language using a computer we want to make sure that we still have meaningful expressions, which can be mapped back to the real world problems.

The main question now is how we can represent knowledge as symbol structures and use that knowledge intelligently to solve problems. How we represent knowledge, using particular *knowledge representation languages*. These are high-level representation formalisms, and can in principle be implemented using a whole range of programming languages. The crucial thing about knowledge representation languages is that they should support *inference*.

(a) Approaches to Knowledge Representation

Broadly speaking, there are three main approaches to knowledge representation in AI. They are the use of logic, structured objects and production systems. The most important is arguably the use of logic. Logic, almost by definition, has a well-defined syntax and semantics, and is concerned with truth preserving inference. However, using logic to represent objects has problems like inefficient and difficulty in representing some common-sense objects.

The idea of structured objects is to represent knowledge as a collection of object attributes and their relations with the object. The most important relations are the *subclass* and *instance* relations. The subclass relation says that one class may be a subset sub-class of another, while the instance relation says that some other individuals belong to same class. We shall use them so that "X subclass Y" means that X is a subclass of Y, not that X has a subclass Y.

Production systems consist of a set of IF-THEN rules, and a working memory. The working memory represents the facts that are currently believed to hold, while the IF-THEN rules typically state that if certain conditions are met, then some action should be taken. If the only action allowed is to add a fact to working memory then rules may be essentially logical implications, but generally greater flexibility is allowed. Production rules capture (relatively) *procedural* knowledge in a simple, modular manner.

(b) Modes of Knowledge Representation

There are many different modes of knowledge representation like object-attribute-value (OAV) triplets, semantic networks, frames, rules, logic programming, and neural networks. Frames are one of the most popular ways of representing non-procedural knowledge in an expert system. In a frame, all the information relevant to a particular concept is stored in a single complex entity, called a frame. Superficially, frames look pretty much like record data structures.

The most common form of logic is that known as *propositional* logic. A proposition, in turn, is a statement that may be either true or false. Propositions may be linked together with various operators (termed logical connectives) such as AND, OR, NOT, and EQUIVAENT.

Neural networks represent mankind's attempt to emulate, in hardware, theories pertaining to the brain. Specifically, it is thought that knowledge is stored in neurons (or, actually, in the connections between neurons).

In the human brain there are more than 10 billion neurons, and each neuron is connected to one or more other neurons, resulting in a massively interconnected network. It is believed that the weightings on each neuron might then represent knowledge to neuron interconnection, which in turn influence the level of strength of the interconnecting impulses.

In this section, the pertinent features of such modes of knowledge representation as object-attribute-value (OAV) triplets, semantic networks, and rules will be described. OAV triplets will be addressed first, not only are they a mode of knowledge representation in themselves, they also form the building blocks of virtually any other approach to knowledge representation.

OAV Triplets

Object-attribute-value triplets provide a particularly convenient way in which to represent certain facts within a knowledge base and may be extended to provide the basis for the representation of heuristic rules. Each OAV triplet is concerned with some specific entity, or object. For example, our object of interest might be an airplane. Associated with every object is a set of attributes that serve to characterize that object. Using the airplane as an example (i.e., as the object), some of its attributes include the following:

- Number of engines
- Type of engine (e.g., jet or prop)
- Type of wing design (e.g., conventional or swept back)

For each attribute, there is an associated value, or set of values. For instance, if we take the C130 military cargo aircraft (known as the Hercules), the number of engines is four, the type of engine is prop, and the wing design is conventional. Values in OAV triplets may be numeric or symbolic. We may list these facts as:

- Number of engines = 4
- Engine type = prop
- Wing design = conventional

In this list, the object itself (i.e., the C130 aircraft) is never explicitly stated. Actually, the above statements represent AV (attribute-value) pairs. However, associated with any AV pair is some object? Thus, any AV pair implies an OAV triplet.

Yet another way to represent an OAV triplet would be through the use of a network representation as indicated in Fig.3. The basic building blocks of a network are its nodes (i.e., the circles) and branches, or edges (i.e., the lines connecting two nodes). In Fig. 3, the object is Pete Jones, the attribute is his income, and the specific value of his income is $50,000.

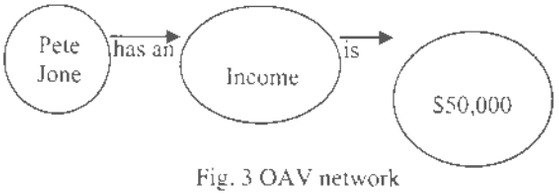

Fig. 3 OAV network

Semantic Net

A semantic net is really just a graph, where the nodes in the graph represent concepts, and the arcs represent binary relationships between concepts. The most important relations between concepts are *subclass* relations between classes and subclasses, and *instance* relations between particular objects and their parent class.

Semantic nets are fine at representing relationships between two objects - but what if we want to represent a relation between three or more objects? In semantic networks we have to view the fact as representing a set of binary relationships between a "giving" (siblings) event and some objects (parents).

A semantic network may be thought of as a network that is composed of multiple OAV triplets in network form as illustrated in Fig.3. However, rather than pertaining to just one attribute for a single object, semantic networks may be used to represent *several* objects, and *several* attributes per object. Returning to our aircraft illustration of the previous section, we might develop a partial semantic network as illustrated in Fig. 4. Here, we note that the C5A is a special type of aircraft (i.e., a large military cargo plane). Further, since the C5A is an aircraft, it *inherits* the properties associated with aircraft in general (e.g., it flies, has wings, and carries people). Such an inheritance property can prove to be of considerable value in the reduction of memory storage requirements. That is, since a C5A is an airplane, there is no need to store, at the C5A node, the fact that it can fly, has wings, and can carry people. Thus, the semantic network scheme provides for a convenient approach for the representation of *associations* between entities.

The OAV-triplet is actually just a restricted subset of semantic networks wherein the only relationships that may be used are those of "is-a" and "has-a." OAV nodes, in turn, may be any of three types: objects, attributes, or values.

To summarize, semantic nets allow us to simply represent knowledge about an object that can be expressed as binary relations. Subclass and instance relations allow us to use *inheritance* to infer new facts/relations from the explicitly represented one.

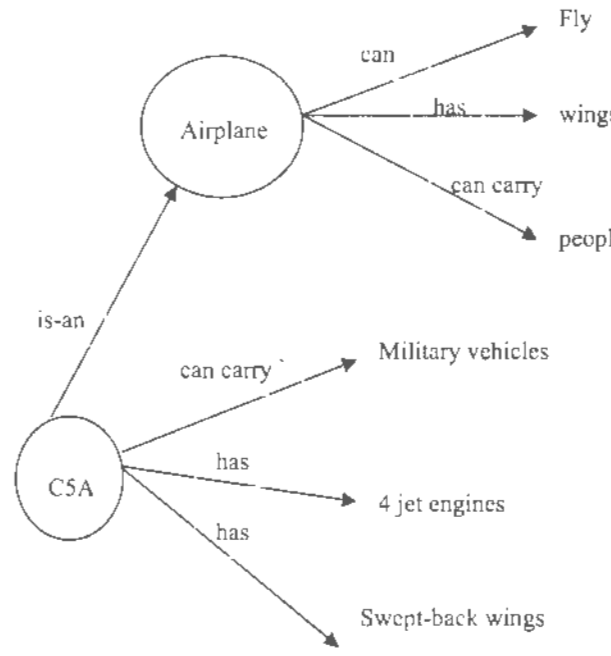

Figure 4 Semantic net

Rule-Based Systems

Instead of representing knowledge in a relatively declarative, static way (as a bunch of objects that are true), rule-based system represent knowledge in terms of a bunch of rules that tell one what one should do or what one could conclude in different situations. A rule-based system consists of a bunch of IF-THEN *rules*, a bunch of *facts*, and some *interpreter* controlling the application of the rules, given the facts.

There are two broad kinds of rule system: *forward chaining* systems, and *backward chaining* systems. In a forward chaining system one start with the initial facts, and keep using the rules to draw conclusions (or take certain actions) given those facts. In a backward chaining system one start with some hypopaper (or goal) one are trying to prove, and keep looking for rules that would allow one to conclude that hypopaper, perhaps setting new subgoals to prove as one go. Forward chaining systems are primarily data-driven, while backward chaining systems are goal-driven.

THE INFERENCE ENGINE

The inference engine is the heart of every expert system. The main purpose of this component is to draw conclusions by applying the abstract knowledge to the concrete knowledge. For Example, in medical diagnosis the symptoms of a given patient (concrete knowledge) are analyzed in the light of the symptoms of all diseases (abstract knowledge).

The conclusions drawn by the inference engine can be based on either deterministic knowledge (clinical tests) or probabilistic knowledge (symptoms). Dealing with uncertain (probabilistic) situations may be considerably more difficult than dealing with certain (deterministic) ones. In most cases some facts (concrete knowledge) are not known with absolute certainty. For example, think of a patient who is not sure about his symptoms. It is also possible to work with abstract knowledge of a non-deterministic type, i.e. where random or fuzzy information is present. The inference engine is also responsible for the propagation of uncertain knowledge. Actually, in probability based expert systems uncertainty propagation is the main task of the inference engine. It enables it to draw conclusions under uncertainty.

The inference engine serves as the inference and control mechanism for the expert system and, as such, is an essential part of the expert system as well as a major factor in the determination of the effectiveness and efficiency of such systems. Inference, in turn, is the process of drawing a conclusion (either intermediate or final) by means of a set of rules, for a specific set of facts, for a given situation. Inference is thus the *knowledge-processing* element of an expert system.

The control responsibilities of the inference engine are those used to determine such matters as

- How to start the inference process
- Which rule to fire if more than one is triggered
- The manner in which the search for a solution is conducted

Like the knowledge base, the inference engine contains rules and facts. However, the rules and facts of the knowledge base pertain to the specific domain of expertise while the rules and facts of the inference engine pertain to the more general control and search strategy employed by the expert system in the development of a solution. These two sets of facts and rules are purposely kept separate in the typical expert system. This is one of the key features of expert systems that serve to differentiate it from heuristic programming.

This separation results in several advantages. First, it permits one to make changes in the knowledge base with minimal impact on the inference engine, and vice versa. Second, it provides for the development and use of expert systems shells.

SEARCH STRATEGIES

The purpose of an expert system is to develop and recommend a proposed solution (or set of alternative solutions) to a given problem. To accomplish this task, the expert system must conduct a *search* for the solution; and it is the responsibility of the inference engine in particular to perform this search in an efficient and effective manner. In the search process, we are faced with a number of alternatives (i.e., potential solutions) and, typically, a variety of constraints.

The two fundamental search strategies employed by an expert system are then forward and backward chaining. Forward chaining proceeds from premises (or data) to conclusions, and is said to be *data driven*. In our inference networks, we would then proceed from left to right. Backward chaining proceeds from a tentative conclusion backward to the premises to determine if the data supports that conclusion. Backward chaining is often called a *goal-driven* approach and proceeds from right to left. Ultimately, both approaches will lead to a conclusion, but their search efficiency is dependent on the nature of the problem faced, that is, on the nature of the inference network associated with the problem.

Specifically, if one has a few premises and many conclusions, then forward chaining is generally the best search strategy. Otherwise, with many premises and relatively few conclusions, we should normally employ backward chaining. The inference networks associated with forward chaining are said to *fan outward* while those for backward chaining *fan inward*.

CONCLUSIONS

Knowledge acquisition phase decides the expert system performance. The major emphasis in knowledge acquisition is the extracting of knowledge about the domain for which the expert system is meant. One has to give a great attention for the rules of thumb used by the domain expert. The quality and performance of the expert system depends to a great extent on the depth of extraction of knowledge and transforming these into rules. The other most important subsystem that requires good insight is the inference engine. Ultimately the expert system is judged by its outcome, conclusion it develops. A convenient inference strategy should be selected and implemented so that the expert system developed would be indispensable tool in the area of interest. One can make a good, reliable expert system if the knowledge engineer follows the steps mentioned and give due attention to important points mentioned.

REFERENCES

[1] Debretsion G.Michael, Counselor Expert System, MSc. Thesis work, June 2000

[2] James P. Ignizio, Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems, McGraw-Hill, 1991

[3] William Siler, PhD, Building fuzzy expert systems, Birmingham, Al 35217, USA, 1997