Unit 10 Expert Systems – I

Structure:

10.1 Introduction
Objectives
10.2 Concept of Expert Systems (ES)
Advantage of ES
Disadvantage of ES
Prominent expert systems and languages
Architectural principles for expert systems
Applications of expert systems
Types of problems solved by expert systems
Expert systems versus other computer based applications
Expert reasoning
10.3 Expert System Features
Goal-driven reasoning
Uncertainty
Data driven reasoning
Data representation
User interface
10.4 Building Blocks of Expert Systems
Knowledge base
Reasoning or Inference Process/Engine
Explaining how or why
Building a knowledge base
The I/O interface
10.5 Rules and Expert Systems
A simple example
Explanation facilities
10.6 Summary
10.7 Terminal Questions
10.8 Answers

10.1 Introduction

In the last unit we discussed conceptual dependency and scripts and so far we have talked a lot about how we can represent knowledge, but not so much about how we can use it to solve real practical problems. Therefore in

this unit, we will look at how some of the techniques discussed so far are used in *expert systems* - systems are computer programs that are derived from *Artificial Intelligence* (AI).

Objectives:

After studying this unit, you should be able to

- explain what expert system is
- list the advantages and disadvantages of expert systems
- list the characteristics of expert reasoning
- list and explain the features of expert systems
- explain principal parts of expert system
- compare and contrast backward chaining and forward chaining.

10.2 Concept of Expert Systems (ES)

Given the number of textbooks, journal articles, and conference publications about expert/knowledge based systems and their application, it is not surprising that there exist a number of different definitions for an expert/knowledge-based system. But we use the following definition: An expert/ or knowledge-based system is a computer program that is designed to mimic the decision-making ability of a decision maker(s) (i.e., expert(s)) in a particular narrow domain of expertise.

Another definition is **"**An expert system is an intelligent computer program that can perform special and difficult task(s) in some field(s) at the level of human experts''.

Expert Systems are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI

Expert systems are most common in a specific problem domain, and is a traditional application and/or subfield of artificial intelligence.So one of the largest area of applications of artificial intelligence is in expert systems, or knowledge based systems as they are often known. It can also be called an information guidance system. Such systems are used for prospecting medical diagnosis or as educational aids. They are also used in engineering and manufacture in the control of robots where they inter-relate with vision systems.

Expert systems attempt to capture the knowledge of a human expert and make it available through a computer system. Expert systems are meant to solve real problems which normally would require a specialized human expert (such as a doctor or a mineralogist). Building an expert system, therefore, first involves extracting the relevant knowledge from the human expert. Such knowledge is often heuristic in nature, based on useful “rules of thumb” rather than absolute certainties. Extracting it from the expert in a way that can be used by a computer is generally a difficult task, requiring its own expertise. A *knowledge engineer* has the job of extracting this knowledge and building the expert system *knowledge base*.

Knowledge acquisition for expert systems is a big area of research, with a wide variety of techniques developed. However, generally it is important to develop an initial prototype, based on information extracted by interviewing the expert, then iteratively refine it based on feedback both from the expert and from potential users of the expert system.

In order to do such iterative development from a prototype it is important that the expert system is written in a way that it can easily be inspected and modified. The system should be able to explain its reasoning (to expert, user and knowledge engineer) and answer questions about the solution process. Updating the system shouldn’t involve rewriting a whole lot of code - just adding or deleting localized chunks of knowledge.

The most widely used knowledge representation scheme for expert systems is rules (sometimes in combination with frame systems). Typically, the rules won't have certain conclusions - there will just be some degree of certainty that the conclusion will hold if the conditions hold. Statistical techniques are used to determine these certainties.

Expert systems have been used to solve a wide range of problems in domains such as medicine, mathematics, engineering, geology, computer science, business, law, defense and education. Within each domain, they have been used to solve problems of different types. The appropriate problem solving technique tends to depend more on the problem type than on the domain.

Expert systems have been some of the most successful applications of A.I. Since these programs must perform in the real world, they encounter important issues for A.I.:

- Lack of sufficient input information
- Probabilistic reasoning

10.2.1 Advantages of ES

The advantages of ES are:

- Provides consistent answers for repetitive decisions, processes and tasks
- Holds and maintains significant levels of information
- Encourages organizations to clarify the logic of their decision-making
- Never “forgets’’ to ask a question, as a human might

10.2.2 Disadvantages of ES

The disadvantages of ES are:

- Lacks common sense needed in some decision making
- Cannot make creative responses as a human expert would in unusual circumstances
- Domain experts not always able to explain their logic and reasoning
- Errors may occur in the knowledge base, and lead to wrong decisions
- Cannot adapt to changing environments, unless knowledge base is changed

10.2.3 Prominent expert systems and languages

These include the following:

- **Mycin** – Diagnose infectious blood diseases and recommend antibiotics (by Stanford University)
- **Dendral** – Analysis of mass spectra
- **Prolog** – Programming language used in the development of expert systems
- **Forth** – Programming language used in the development of expert systems
- **ART** – An early general-purpose programming language used in the development of expert systems
- **CADUCEUS** (expert system) – Blood-borne infectious bacteria
- **CLIPS –** Programming language used in the development of expert systems

- **Drools –** An open source offering from JBOSS labs
- **Dipmeter Advisor** – Analysis of data gathered during oil exploration
- **Jess** – Java Expert System Shell. A CLIPS engine implemented in Java used in the development of expert systems
- **KnowledgeBench** – expert system for building new product development applications
- **Knowledge Engineering Environment** – e-mycin derivative that implemented ATMS
- **MQL 4** – MetaQuotes Language 4, a customized language for financial strategy programming
- **NETeXPERT** – A mission-critical Operational Support Systems framework with rules, policies, object modeling, and adapters for Network Operations Center automation
- **NEXPERT Object** – An early general-purpose commercial backwards-chaining inference engine used in the development of expert systems
- **R1/Xcon** – Order processing
- **SHINE Real-time Expert System** – Spacecraft Health INference Engine
- **STD Wizard** – Expert system for recommending medical screening tests
- **PyKe** – Pyke is a knowledge-based inference engine (expert system)

10.2.4 Architectural principles for expert systems

The Architectural principles for expert systems are:

1. Knowledge is power
2. Knowledge is often inexact & incomplete
3. Knowledge is often poorly specified
4. Expert systems need to be flexible
5. Expert systems need to be transparent
6. Separate inference engine and knowledge base level
7. Use uniform “fact’’ representation
8. Keep inference engine simple
9. Explicit redundancy
10. Amateurs experts slowly

10.2.5 Applications of expert systems

Expert systems are designed and created to facilitate tasks in the fields of accounting, medicine, process control, financial service, production, human resources etc.

A good example of application of expert systems in banking area is expert systems for mortgages. Loan departments are interested in expert systems for mortgages because of the growing cost of labor which makes the handling and acceptance of relatively small loans less profitable. They also see in the application of expert systems a possibility for standardized, efficient handling of mortgage loan, and appreciate that for the acceptance of mortgages there are hard and fast rules which do not always exist with other types of loans.

While expert systems have distinguished themselves in AI research in finding practical application, their application has been limited. Expert systems are notoriously narrow in their domain of knowledge – as an amusing example, a researcher used the “skin disease’’ expert system to diagnose his rust bucket car as likely to have developed measles–and the systems were thus prone to making errors that humans would easily spot.

An example and a good demonstration of the limitations of, an expert system used by many people is the Microsoft Windows operating system troubleshooting software located in the “help’’ section in the taskbar menu. Obtaining expert / technical operating system support is often difficult for individuals not closely involved with the development of the operating system. Microsoft has designed its expert system to provide solutions, advice, and suggestions to common errors encountered throughout using the operating systems.

Another 1970s and 1980s application of expert systems – which we today would simply call AI – was in computer games. For example, the computer baseball games Earl Weaver Baseball and Tony La Russa Baseball each had highly detailed simulations of the game strategies of those two baseball managers. When a human played the game against the computer, the computer queried the Earl Weaver or Tony La Russa Expert System for a decision on what strategy to follow. Even those choices where some randomness was part of the natural system (such as when to throw a surprise pitch-out to try to trick a runner trying to steal a base) were decided based on probabilities supplied by Weaver or La Russa. Today we would simply say that “the game’s AI provided the opposing manager’s strategy.’’

10.2.6 Types of problems solved by expert systems

Expert systems are designed to carry the intelligence and information found in the intellect of experts and provide this knowledge to other members of the organization for problem-solving purposes.

Typically, the problems to be solved are of the sort that would normally be tackled by a medical or other professional. Real experts in the problem domain are asked to provide “rules of thumb’’ on how they evaluate the problems, either explicitly with the aid of experienced systems developers, or sometimes implicitly, by getting such experts to evaluate test cases and using computer programs to examine the test data and (in a strictly limited manner) derive rules from that. Generally, expert systems are used for problems for which there is no single “correct’’ solution which can be encoded in a conventional algorithm.

10.2.7 Expert systems versus other computer based applications

Expert systems tend to be more effective than other computer based applications, because they:

- may combine the knowledge of many experts in a specific field,
- can store an unlimited amount of information, and works much faster, than a human,
- are available 24 hours a day, and can be used at a distance over a network,
- are able to explain their information requests and suggestions,
- can process client’s uncertain responses and, by combining several pieces of uncertain information, may still be able to make strong recommendations,
- can accumulate the knowledge of high level employees for any company, which is especially useful when the company needs to fire them due to worsened market conditions.

10.2.8 Expert Reasoning

Expert reasoning typically has special characteristics:

- Use of specialized representations appropriate to the domain and specialized problem-solving methods based on those representations.
- Translation of observables into specialized terminology and representations (e.g., “person has turned blue’’ into “patient is cyanotic’’).

- Use of empirical rules of thumb (e.g., “to blow out a tree stump, use one stick of dynamite per 4 inches of stump diameter’’.)
- Use of empirical correlations (e.g., certain bacteria have been observed to be likely to cause infection in burn patients).
- Use of “incidental” facts to discriminate cases (e.g., “a snake that swims with its head out of the water is a water moccasin’’). Such discrimination depends on the sparseness of the domain (only certain snakes are possible).

Self Assessment Questions

1. Expert systems attempt to capture the knowledge of a human expert and make it available through a _____ system.
2. The most widely used knowledge representation scheme for expert systems is _____.
3. _____ is the Programming language used in the development of expert systems.
4. A good example of application of expert systems in banking area is expert systems for _____.

10.3 Expert System Features

Now I will tell you the important features of Expert System. There are a number of features which are commonly used in expert systems. The major features are:

- *Goal driven reasoning or backward chaining* – an inference technique which uses IF THEN rules to repetitively break a goal into smaller sub-goals which are easier to prove;
- *Coping with uncertainty* – the ability of the system to reason with rules and data which are not precisely known;
- *Data driven reasoning or forward chaining* – an inference technique which uses IF THEN rules to deduce a problem solution from initial data;
- *Data representation* – the way in which the problem specific data in the system is stored and accessed;
- *User interface* – that portion of the code which creates an easy to use system;
- *Explanations* – the ability of the system to explain the reasoning process that it used to reach a recommendation.

10.3.1 Goal-driven reasoning

Goal-driven reasoning, or backward chaining, is an efficient way to solve problems that can be modeled as “structured selection’’ problems. For example, an identification problem falls in this category. Diagnostic systems also fit this model, since the aim of the system is to pick the correct diagnosis.

The knowledge is structured in rules which describe how each of the possibilities might be selected. The rule breaks the problem into sub-problems. For example, the following top level rules are in a system which identifies birds.

IF
family is albatross and
color is white

THEN
bird is laysan albatross.

IF
family is albatross and
color is dark

THEN
bird is black footed albatross.

The system would try all of the rules which gave information satisfying the goal of identifying the bird. Each would trigger sub-goals. In the case of these two rules, the sub-goals of determining the family and the color would be pursued. The following rule is one that satisfies the family sub-goal:

IF
order is tubenose and
size large and
wings long narrow

THEN
family is albatross.

The sub-goals of determining color, size, and wings would be satisfied by asking the user. By having the lowest level sub-goal satisfied or denied by the user, the system effectively carries on a dialog with the user. The user

sees the system asking questions and responding to answers as it attempts to find the rule which correctly identifies the bird.

10.3.2 Uncertainty

Many times in structured selection problems the final answer is not known with complete certainty. The expert's rules might be vague, and the user might be unsure of answers to questions. For expert systems to work in the real world they must also be able to deal with uncertainty. One of the simplest schemes is to associate a numeric value with each piece of information in the system. The numeric value represents the certainty with which the information is known. There are numerous ways in which these numbers can be defined, and how they are combined during the inference process.

10.3.3 Data driven reasoning

For many problems it is not possible to enumerate all of the possible answers before hand and have the system select the correct one. For example, configuration problems fall in to this category. These systems might put components in a computer, design circuit boards, or lay out office space. Since the inputs vary and can be combined in an almost infinite number of ways, the goal driven approach will not work.

The data driven approach, or forward chaining, uses rules similar to those used for backward chaining, however, the inference process is different. The system keeps track of the current state of problem solution and looks for rules which will move that state closer to a final solution.

A system to layout living room furniture would begin with a probable state consisting of a number of unplaced pieces of furniture. Various rules would be responsible for placing the furniture in the room, thus changing the problem state. When all of the furniture was placed, the system would be finished, and the output would be the final state. Here is a rule from such a system which places the television opposite the couch.

IF
unplaced tv and
couch on wall(X) and
wall(Y) opposite wall(X)

THEN
place tv on wall(Y).

This rule would take a problem state with an unplaced television and transform it to a state that had the television placed on the opposite wall from the couch. Since the television is now placed, this rule will not fire again. Other rules for other furniture will fire until the furniture arrangement task is finished.

Note that for a data driven system, the system must be initially populated with data, in contrast to the goal driven system which gathers data as it needs it. Figure 10.1 illustrates the difference between forward and backward chaining systems for two simplified rules.

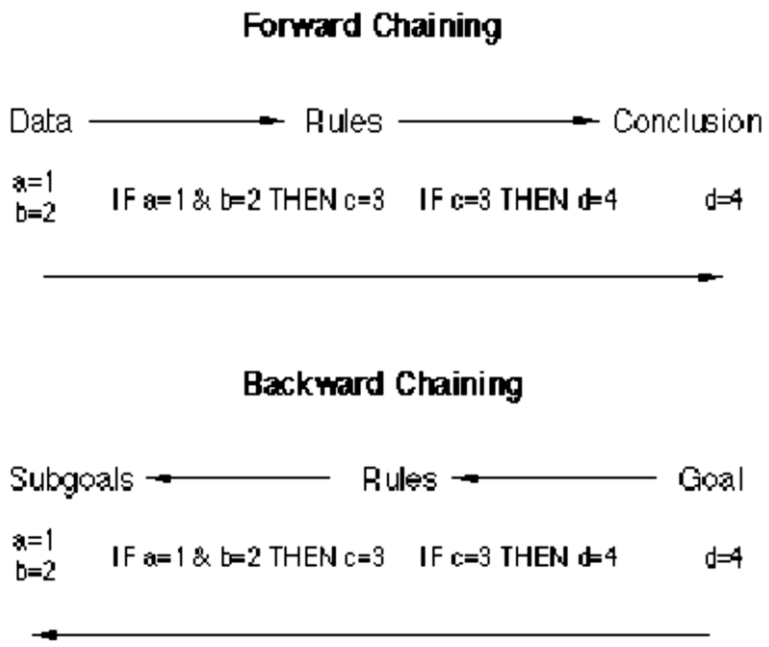

Figure 10.1: Difference between forward and backward chaining

The forward chaining system starts with the data of **a=1** and **b=2** and uses the rules to derive **d=4**. The backward chaining system starts with the goal of finding a value for **d** and uses the two rules to reduce that to the problem of finding values for **a** and **b**.

10.3.4 Data representation

For all rule based systems, the rules refer to data. The data representation can be simple or complex, depending on the problem. The four levels described in this section are illustrated in figure 10.2.

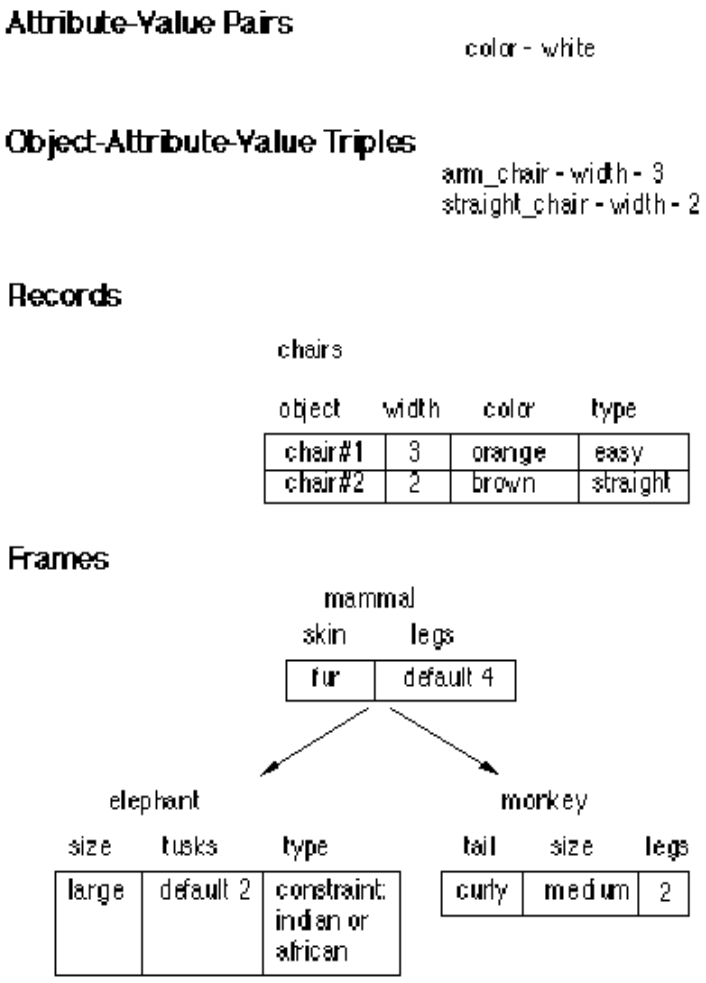

Figure 10.2: Four levels of data representation

The most fundamental scheme uses attribute-value pairs as seen in the rules for identifying birds. Examples are color-white, and size-large.

When a system is reasoning about multiple objects, it is necessary to include the object as well as the attribute-value. Once there are objects in the system, they each might have multiple attributes.

Frames are a more complex way of storing objects and their attribute-values. Frames add intelligence to the data representation, and allow objects to inherit values from other objects.

In a furniture placement system each piece of furniture can inherit default values for length. When the piece is placed, demons are activated which automatically adjust the available space where the item was placed.

10.3.5 User interface

The acceptability of an expert system depends to a great extent on the quality of the user interface. The easiest to implement interfaces communicate with the user through a scrolling dialog as illustrated in figure 10.3. The user can enter commands, and respond to questions. The system responds to commands, and asks questions during the inferencing process.

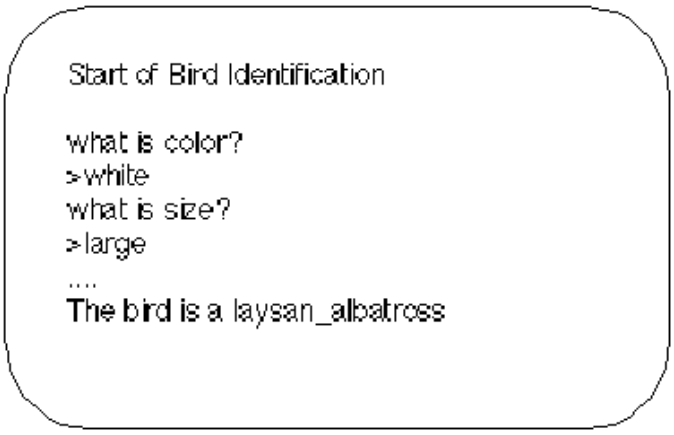

Figure 10.3: Scrolling dialog user interface

More advanced interfaces make heavy use of pop-up menus, windows, mice, and similar techniques as shown in figure 10.4. If the machine supports it, graphics can also be a powerful tool for communicating with the user. This is especially true for the development interface which is used by the knowledge engineer in building the system.

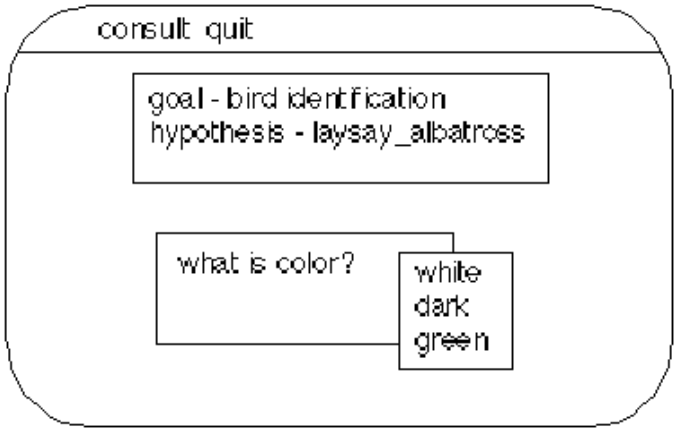

Figure 10.4: Window and menu user interface

Explanations

One of the more interesting features of expert systems is their ability to explain themselves. Given that the system knows which rules were used during the inference process, it is possible for the system to provide those rules to the user as a means for explaining the results.

This type of explanation can be very dramatic for some systems such as the bird identification system. It could report that it knew the bird was a black footed albatross because it knew it was dark colored and an albatross. It could similarly justify how it knew it was an albatross.

Explanations are always of extreme value to the knowledge engineer. They are the program traces for knowledge bases. By looking at explanations the knowledge engineer can see how the system is behaving, and how the rules and data are interacting. This is an invaluable diagnostic tool during development.

Self Assessment Questions

5. Which are the more complex way of storing objects and their attribute-values?

   _____

6. How data driven approach is different from backward chaining?

   _____

10.4 Building Blocks of Expert Systems

Every expert system consists of two principal parts:

- the knowledge base; and
- the reasoning, or inference, engine.

10.4.1 Knowledge base

The *knowledge base* of expert systems contains both factual and heuristic knowledge. *Factual knowledge* is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field. *Heuristic knowledge* is the less rigorous, more experiential, more judgmental knowledge of performance. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the "art of good guessing.'' *Knowledge representation* formalizes and organizes the knowledge. One widely used representation is the *production rule*, or simply *rule*. A rule consists of an IF part and a THEN part (also called a *condition* and an *action*). The IF part lists a set of conditions in some logical combination. The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving action taken. Expert systems whose knowledge is represented in rule form are called *rule-based systems*.

Another widely used representation, called the *unit* (also known as *frame*, *schema*, or *list structure*) is based upon a more passive view of knowledge. The unit is an assemblage of associated symbolic knowledge about an entity to be represented. Typically, a unit consists of a list of properties of the entity and associated values for those properties.

10.4.2 Reasoning, or Inference Process/Engine

The *problem-solving model,* or *paradigm*, organizes and controls the steps taken to solve the problem. One common but powerful paradigm involves chaining of IF-THEN rules to form a line of reasoning. If the chaining starts from a set of conditions and moves toward some conclusion, the method is called *forward chaining*. If the conclusion is known (for example, a goal to be achieved) but the path to that conclusion is not known, then reasoning backwards is called for, and the method is *backward chaining*. These problem-solving methods are built into program modules called *inference*

engines or *inference procedures* that manipulate and use knowledge in the knowledge base to form a line of reasoning. The *knowledge base* an expert uses is what he learned at school, from colleagues, and from years of experience. Presumably the more experience he has, the larger his store of knowledge.

The inference process is carried out recursively in three stages: (1) match, (2) select, and (3) execute. During the match stage, the contents of working memory are compared to facts and rules contained in knowledge base. When consistent matches found, the corresponding rules are placed in a conflict set. To find an appropriate and consistent match, substitution may be required. Once all the matched rules have been added to the conflict set during a given cycle, one of the rules is selected for execution. The criteria for selection may be most recent use, rule condition specificity, (the number of conjuncts on the left), or simply the smallest rule number. The selected rule is then executed and the right hand side or action part of the rule is then carried out. Figure 10.5 illustrates this match-select-execute cycle.

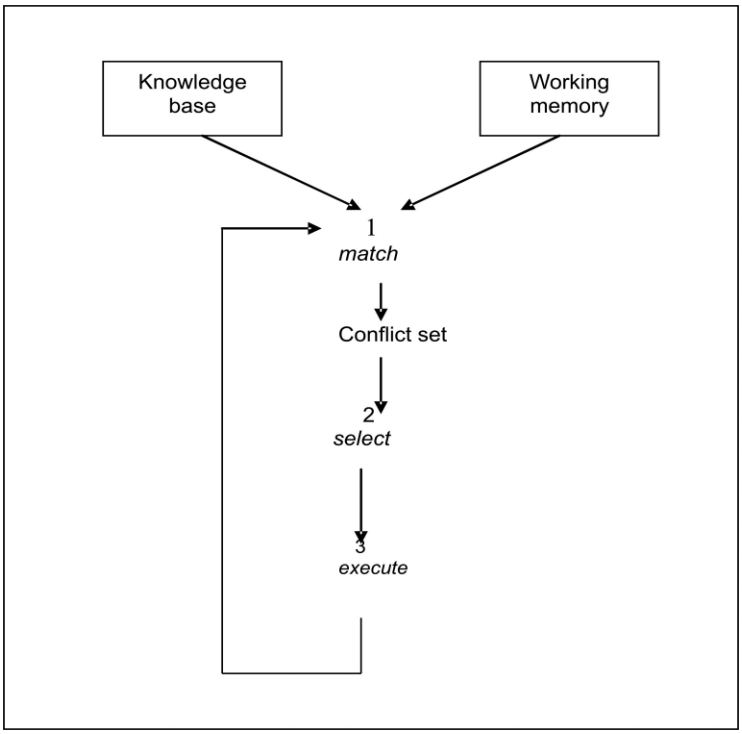

Figure 10.5: The production system inference cycle

10.4.3 Explaining how or why

The explanation module provides the user with an explanation of the reasoning process when requested. This is done in response to a how query or a why query.

To respond to a how query, the explanation module traces the chain of rules fired during a consultation with the user. The sequence of rules that led to the conclusion is then printed for the user in an easy to understand human-language style. This permits the user to actually see the reasoning process followed by the system in arriving at the conclusion. If the user does not agree with the reasoning steps presented, they may be changed using an editor.

To respond to a why query, the explanation module must be able to explain why certain information is needed by the inference engine to complete a step in the reasoning process before it can proceed. Though an expert system consists primarily of a knowledge base and an inference engine, a couple of other features are worth mentioning: reasoning with uncertainty, and explanation of the line of reasoning. Knowledge is almost always incomplete and uncertain. To deal with uncertain knowledge, a rule may have associated with it a *confidence factor* or a weight. The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called *reasoning with uncertainty*. An important subclass of methods for reasoning with uncertainty is called “fuzzy logic,’’ and the systems that use them are known as “fuzzy systems.’’ Because an expert system uses uncertain or heuristic knowledge (as we humans do) its credibility is often in question (as is the case with humans). When an answer to a problem is questionable, we tend to want to know the rationale. If the rationale seems plausible, we tend to believe the answer. So it is with expert systems.

10.4.4 Building a knowledge base

The editor is used by developers to create new rules for addition to the knowledge base, to delete outmoded rules, or to modify existing rules in some way. Some of the most sophisticated expert system editors provide the user with features not found in typical text editors, such as the ability to perform some type of consistency tests for newly created rules, to add missing conditions to a rule, or to reformat a newly created rule. Such

systems also prompt the user for missing information, and provide other general guidance in the KB creation process. One of the most difficult task in creating and maintaining production systems is the building and maintaining of a consistent but complete set of rules. This should be done without adding redundant or unnecessary rules. Building a knowledge base requires careful planning, accounting and organization of the knowledge structures. It also requires thorough validation and verification of the completed knowledge base, operations which have yet to be perfected. An “intelligent’’ editor can greatly simplify the process of building a knowledge base.

10.4.5 The I/O interface

The I/O interface permits the user to communicate with the system in a more natural way by permitting the use of simple selection menus or the use of a restricted language which is close to a natural language. The most important ingredient in any expert system is knowledge. The power of expert systems resides in the specific, high-quality knowledge they contain about task domains. AI researchers will continue to explore and add to the current repertoire of knowledge representation and reasoning methods. But in knowledge resides the power. Because of the importance of knowledge in expert systems and because the current knowledge acquisition method is slow and tedious, much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure.

Self Assessment Questions

7. _____ is that knowledge of the task domain that is widely shared.
8. The inference process is carried out recursively in _____ stages.
9. What is the most important ingredient of expert system?

   _____

10.5 Rules and Expert Systems

In this section we will show how expert systems based on IF-THEN rules work, and present a very simple expert system shell.

Rule-based systems can be either *goal driven* using *backward chaining* to test whether some hypothesis is true, or *data driven*, using *forward chaining* to draw new conclusions from existing data. Expert systems may use either or both strategies, but the most common is probably the goal driven/ backward chaining strategy. In a simple goal-driven rule-based expert system there are often a set of possible solutions to the problem - maybe these are a set of illnesses that the patient might have. The expert system will consider each hypothesized solution (e.g., has_cold(fred)) and try to prove whether or not it might be the case. Sometimes it won"t be able to prove or disprove something from the data initially supplied by the user, so it will ask the user some questions (e.g., "have you got a headache?''). Using any initial data plus answers to these questions it should be able to conclude which of the possible solutions to the problem is the right one. We explore it using:

- A Simple Example
- Explanation facilities
- More Complex Systems

10.5.1 A simple example

This is much better explained through a simple example. Suppose that we have the following rules:

1) IF engine_getting_petrol
   AND engine_turns_over
   THEN problem_with_spark_plugs
2) IF NOT engine_turns_over
   AND NOT lights_come_on
   THEN problem_with_battery
3) IF NOT engine_turns_over
   AND lights_come_on
   THEN problem_with_starter
4) IF petrol_in_fuel_tank
   THEN engine_getting_petrol

Our problem is to work out what's wrong with our car given some observable symptoms. There are three possible problems with the car: problem_with_spark_plugs, problem_with_battery, problem_with_starter. We'll assume that we have been provided with no initial facts about the observable symptoms.

In the simplest goal-directed system we would try to prove each hypothesised problem (with the car) in turn. First the system would try to prove “problem_with_spark_plugs’’. Rule 1 is potentially useful, so the system would set the new goals of proving “engine_getting_petrol’’ and “engine_turns_over’’. Trying to prove the first of these, rule 4 can be used, with new goal of proving “petrol_in_fuel_tank.’’ There are no rules which conclude this (and the system doesn't already know the answer), so the system will ask the user:

Is it true that there’s petrol in the fuel tank?

Let’s say that the answer is yes. This answer would be recorded, so that the user doesn’t get asked the same question again. Anyway, the system now has proved that the engine is getting petrol, so now wants to find out if the engine turns over. As the system doesn’t yet know whether this is the case, and as there are no rules which conclude this, the user will be asked:
Is it true that the engine turns over?

Let’s say this time the answer is no. There are no other rules which can be used to prove “problem_with_spark_plugs’’ so the system will conclude that this is not the solution to the problem, and will consider the next hypothesis: problem–with–battery. It is true that the engine does not turn over (the user has just said that), so all it has to prove is that the lights don't come on. It will ask the user

Is it true that the lights come on?

Suppose the answer is no. It has now proved that the problem is with the battery. Some systems might stop there, but usually there might be more than one solution, (e.g., more than one fault with the car), or it will be uncertain which of various solutions is the right one. So usually all hypotheses are considered. It will try to prove “problem_with_starter’’, but given the existing data (the lights come on) the proof will fail, so the system will conclude that the problem is with the battery. A complete interaction with our very simple system might be:

System: Is it true that there's petrol in the fuel tank?
User: Yes.
System: Is it true that the engine turns over?
User: No.

System: Is it true that the lights come on?
User: No.
System: I conclude that there is a problem with battery.

Note that in general, solving problems using backward chaining involves *searching* through all the possible ways of proving the hypothesis, systematically checking each of them. A common way of doing this search is the same as in Prolog - depth first search with backtracking.

10.5.2 Explanation facilities

Most expert systems have explanation facilities that allow the user to ask *why* it asked some question, and *how* it reached some conclusion. These questions are answered by referring to the system goals, the rules being used, and any existing problem data. To illustrate the sorts of facilities commonly provided, we'll elaborate on our simple example above, starting with a simple example dialogue involving *why* and *how* questions and explanations.

System: Is there gas in the fuel tank?
User: Yes.
System: Is it true that the engine turns over?
User: No.
System: Is it true that the lights come on?
User: Why?
System: I am trying to prove: problem with battery.
It has been established that it is not the case that the engine turns over.
Therefore if it is not the case that the lights come on then there is a problem with battery.
Is it true that the lights come on?
User: No.
System: I conclude that there is a problem with the battery.
User: How?
System: This follows from Rule 2:
IF NOT engine_turns_overAND ...
NOT engine_turns_overwas given by the user.
NOT lights_come_onwas given by the user.

Providing such an explanation facility involves, at least, recording what rules are used in drawing conclusions, and using these records to compose

explanations like the ones above. Explanation facilities in expert systems are often not widely used, and where used not viewed as acceptable by their users. There are a whole lot of reasons for this, motivating current research in the area. One reason is that the explanations just reference the "surface'' knowledge encoded in the rules, rather than the "deep'' knowledge about the domain which originally motivated the rules (but which is usually not represented). So, the system will say that it concluded X because of rule23, but not explain what rule23 is all about. (In the above example, maybe the user needs to understand that both the lights and the starter use the battery, which is the underlying rationale for the second rule in our example). Another stated reason for the frequent failure of explanation facilities is the fact that, if the user fails to understand or accept the explanation, the system can't re-explain in another way (as people can). Explanation generation is a fairly large (and fascinating) area of research, concerned with effective communication.

More Complex Systems

In general, there would be hundreds or thousands of rules in the system, and each rule would be considerably more complex. Rules will almost certainly involve patterns with variables (e.g., age(Patient, X)), rather than just simple unstructured propositions (e.g., age_fred_23)) and will usually have certainty factors attached to them (described later).

However, although real systems will be much more complex than the above, many use the same basic reasoning and explanation procedures. One early system which used this basic approach is MYCIN, a system for diagnosing blood disorders.

Self Assessment Questions

10. Solving problems using backward chaining involves _____ through all the possible ways of proving the hypothesis, systematically checking each of them.
11. Most expert systems have explanation facilities that allow the user to ask *why* it asked some question, and *how* it reached some conclusion. (State True or False)
12. Rules will almost certainly involve _____ with variables.

10.6 Summary

In this unit, we discussed the concept of Expert Systems, architectural Principles and applications of expert systems and types of problems solved by expert systems. We also discussed expert system features and building blocks of ES, and rules. One of the largest area of applications of artificial intelligence is in expert systems, or knowledge based systems. Expert systems are designed and created to facilitate tasks in the fields of accounting, medicine, process control, financial service, production, human resources etc. The knowledge base of expert systems contains both factual and heuristic knowledge. Rule-based systems can be either *goal driven* using *backward chaining* to test whether some hypothesis is true, or data driven, using forward chaining to draw new conclusions from existing data.

10.7 Terminal Questions

1. What is an expert system? Explain briefly.
2. What are the advantages and disadvantages of expert systems?
3. What are the expert system's architectural principles?
4. List and explain briefly the features of expert systems.
5. Write a note on knowledge base.
6. Compare and contrast backward chaining and forward chaining.

10.8 Answers

Self Assessment Questions

1. computer
2. rules
3. CLIPS
4. mortgages
5. Frames
6. Inference process is different
7. Factual knowledge
8. three
9. knowledge
10. searching
11. True
12. patterns

Terminal Questions

1. An expert/ or knowledge-based system is a computer program that is designed to mimic the decision-making ability of a decision maker(s) (i.e., expert(s)) in a particular narrow domain of expertise.
   (Refer section 10.2 for detail)
2. Advantage: Provides consistent answers for repetitive decisions, processes and tasks
   Disadvantage: Lacks common sense needed in some decision making.
   (Refer sub-sections 10.2.1 and 10.2.2 for various others advantages and disadvantages)
3. Architectural Principles for Expert Systems are:
   - Knowledge is power
   - Knowledge is often inexact & incomplete

   (Refer sub-section 10.2.4 for detail)
4. The major features of expert systems are:
   - Goal driven reasoning or backward chaining (Refer section 10.3 for detail)
5. The knowledge base of expert systems contains both factual and heuristic knowledge. (Refer sub-section 10.4.1 for detail)
6. Forward chaining is a method in which If the chaining starts from a set of conditions and moves toward some conclusion. (Refer sub-section 10.4.2 for detail)