Expert System

## Contents:

* Introduction
* Expert System, Feature, Characteristics, Development, Architecture
* Goals and Basic Activities and Advantages
* Stages in development of Expert system
* Probability Based Expert System
* Expert System Tools

# Introduction

Expert systems are computer programme built for commercial application using the programming techniques of Artificial Intelligence, especially those developed for problem solving. Expert System have been built for various purposes including medical diagnosis, electronic fault finding, and mineral prospecting and computer system configuration. The *inference engine* of an expert system operates on the knowledge base and produces inferences. For the purpose of constructing expert system, *knowledge acquisition* is used to obtain domain knowledge about an application.

# Expert System

Expert system has been most important branch of Artificial Intelligence. It was perceived as in early times that the human brain makes decisions of the “YES” and “NO” kind, i.e. true or false. In 1854, George Boole first published his article *Investigation on the laws of thought,* which gave birth to Boolean algebra and set theory. Advent of electronic circuits and solid-state integrated circuits helped bring, about the modern era of Von Neumann type sequential digital computation. Digital computers were known as “intelligent” machine owing it to their capability to process human thought i.e. yes(1) or no(0). Since, 1960s, it has been believed that computers have severe limitations only in solving algorithm-type problems. To tackle this problem, an entirely new way of programming a computer, which closely matches the human logical thinking process, called the “Expert System” has developed.

Expert system provide the following important features:

* Facility for non-expert personnel to solve problems that require some expertise
* Speedy solutions
* Reliable solutions
* Cost reduction
* Elimination of uncomfortable and monotonous operations
* Power to manage without human experts
* Wider access to knowledge

In addition to the advantages given above, the use of exert system is especially recommended when

* Human experts are difficult to find
* Human experts are expensive
* Knowledge improvement is needed
* Knowledge is difficult to acquire and is based on rules that can only be learnt through experience
* The available information is poor, partial, incomplete
* Problems are incompletely defined
* There is lack of knowledge among all those who need it
* The problem is subject to rapidly changing legal rules and codes

The human being may have multiple areas of expertise. Ex: a power system professional who has the special or domain expertise required for diagnosing power transformers. The professional can apply power and hence voltage to the input terminal and measure the output voltage and current. Based on this information, the power system professional with his/her knowledge or expertise can detect defective or faulty parts. He has learned this and acquired this knowledge through experience over prolonged period of time and education. The question is: Is it possible to implant the same knowledge in computer program so that it can replace the human expert? The answer is “YES”.

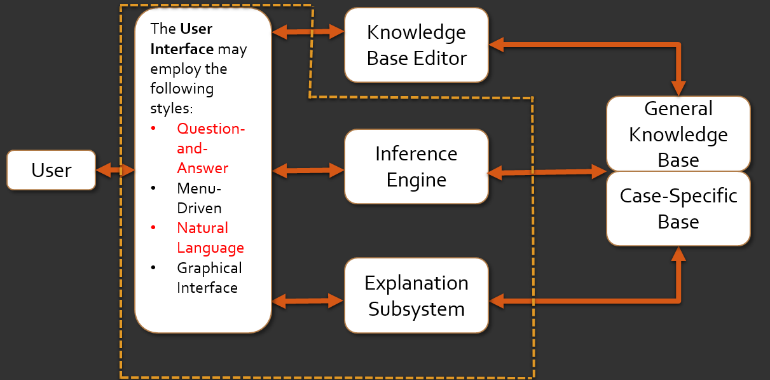


Fig: Development of Expert System

Owing to the heuristic, knowledge- intensive nature of exert level problem solving, expert system are generally:

* Open to inspection
* Easily modifiable
* Heuristic, in using (often imperfect) knowledge to obtain solution

An expert system is open to inspection in that the user may, at any time during programme execution, inspect the state of its reasoning and determine the specific choices and decisions that the programme is making. This is mandatory because if human expert is to accept the recommendation from computer then he/she must be satisfied with the solution. “The computer said so” is not sufficient reason to follow the instruction.

When the solution is open to inspection, we can evaluate every decision taken during the solution process, allowing for partial agreement and the addition of new information or rules to improve the performance. This plays an essential role in refinement of a knowledge base.

The heuristic nature of expert problem-solving knowledge creates problems in the evaluation of programme performance. Although we know that heuristic approach might fail, it is not often clear exactly how often programme must be correct to be accepted: should it be 90% of the time or 80% of the time or something else. Perhaps the best way to evaluate a programme is to compare its results to those obtained by human experts in the same area.

# Features

We can represent the knowledge of an expert system using rules and objects. We normally represent knowledge in the form of IF-THEN rules as follows:

*IF the load demand is medium, THEN the system is reliable.*

The knowledge Base consists the fact that the load demand is medium, it will match the condition of the rule. The rule is thus satisfied and we can conclude that the system is reliable.

We can built many significant expert systems by expressing the knowledge of experts in the form of rules. Rules can pattern-match on objects as well as facts. A wide range of Knowledge Base expert systems have been recently developed. The fundamental entities which plays important role in development of Expert system are: Knowledge engineer, Domain Expert and Human Expert.

*An expert system is a collection of programmes or computer software that solve problems in the domain of interest.*

An expert system consists of both problem solving component and a support component. The process of building expert system is called knowledge engineering and is done by knowledge engineer. The knowledge engineer is a human with the background of computer science and AI, and knows how to build an expert system. Knowledge Engineer also decides how to represent the knowledge in an expert system and help the programmer to write code.

Knowledge Engineering is the acquisition of knowledge from a human expert or any other source.

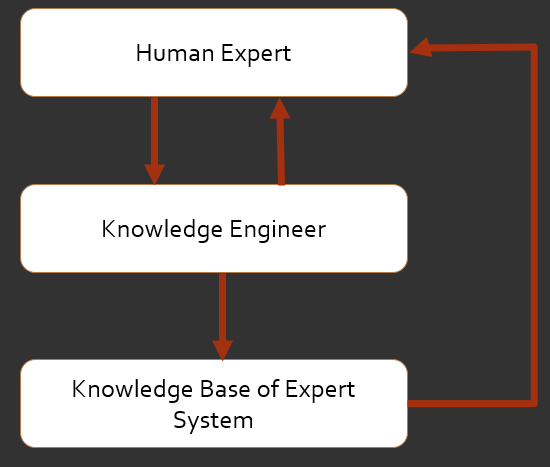


Fig: Development of an Expert System

Knowledge Engineer creates a relationship with the human expert in order to elicit knowledge from him/her. He then codes all the knowledge contained in the knowledge base. The expert then evaluates the exert system and gives a report to the knowledge engineer. The process continues until the performance of the system is found satisfactory by the expert. Expert system must have explanation facility. Some expert system learn rules through the rule induction, in which rules are created using data table.

A practical limitation of many expert systems is lack of General knowledge. Expert system do not really understand the underlying cause and effect in the system.

# Characteristics

The basic characteristics of Expert system are:

* High Performance: The quality of the advice given by the system should be very high. The system must be in a position to respond at all level of competency equal to or better than that of an expert in the field.
* Expertise: Real expert not only produce good solutions but also find them quickly. So, an expert system must be skillful in applying its knowledge to produce solutions both efficiently and effectively using the intelligence human expert normally use to eliminate wasteful or unnecessary calculations. The expert system should be robust.
* Adequate Response time: The system should be designed in such a way that it is able to perform within a reasonable amount of time, comparable to or better than the time taken by human expert to reach a decision. The time constraints placed on the performance of the expert system may be especially severe in the case of real-time systems, when a response is required within a certain time interval.
* Good Reliability: The expert system must be as reliable as human experts.
* Self-knowledge: The explanation facility is the knowledge that explains how a system arrived at its answers. The ability to examine their reasoning process and explain their operation is one of the most innovative and important qualities of expert systems. Self-knowledge is considered one of the important characteristics of expert systems because:

1. User tend to have more faith in the result and more confidence in the system
2. System development is faster since the system is very easy to compile and debug
3. The assumptions underlying the system’s operation are explicit rather than implicit
4. The effect of a change on the system operation is easily tested and predicted

* Understandable: The system should be able to explain the steps of reasoning while executing. The expert system have an explanation capability similar to the reasoning ability of human experts.
* Justification: This allows the user to ask the expert system to justify the solution or advice provided by it. Normally, expert system justify their answers or advice by explaining their reasoning. If the system is rule based one, it provides to the user all the rules and facts it has used to achieve its answer.
* Flexibility: As the expert system has a large amount of knowledge, it is important for it to have an efficient mechanism for adding, changing and deleting knowledge. Rule-based system have become popular only due to efficiency and modular capability of rules.

Depending on the kind of expert system, an explanation facility may be simple or elaborate. Many expert systems will do the following:

1. List all the reasons for and against a particular hypothesis. A hypothesis is a goal that is to be proved, such as, “a patient have viral fever” in a medical diagnostic expert system. We can also define hypothesis as a fact whose truth is in doubt and must be proved.
2. List all the entire hypothesis that may explain the observed evidence.
3. Explain all the consequences of a hypothesis.
4. Predict what will occur if the hypothesis is true
5. Justify that questions asked by the programme to the user for further information

# Development

AI has branches related to speech, vision, robotics and natural language processing. Once of the major root of expert system is the area of human information processing called cognitive science. Cognition is the study of how human process information i.e. how people think while solving problems.

The study of cognition is fundamental if we want computers to emulate human experts. Normally, experts can’t explain how a problem is solved by them but can provide solution. If expert cannot explain how a problem is solved, it is not possible to encode the knowledge in an expert system based on explicit knowledge. The only option then is for the programmes to themselves learn how to emulate the expert. These programmes are based on induction.

Late 1950s and early 1960s: number of programme were written with the goal of general problem solving. Newell and Simon created General problem solver which demonstrated that much of human problem solving or cognition could be expressed using IF-THEN type condition rules.

A rule that corresponds to a small modular collection of knowledge is called chunk. Ex*: IF you want to get good marks and a good job THEN work hard*

Newman and Simon popularized the use of rules to represent human knowledge and showed how reasoning could be done with the help of rules. By the early 1970s, it has become apparent that domain knowledge was the key to building machine problem solver that could function at the level of human expert. Although methods for reasoning are important, studies have shown that experts do not primarily rely on reasoning in problem. Reasoning play vital role in problem solving but experts rely in the vast knowledge heuristics and experience they have built over years. Early attempt at building powerful problem solver based only on reasoning have shown that such systems are crippled. Today, an expert system is considered as an alternative to conventional algorithmic programming.

By late 1970s, three concepts basic to expert system today has converged. These are, a) Rules b) Shell c) Knowledge.

By 1980s, companies started to bring expert system commercially. Like Automated Reasoning Tools by Inference Corp., Knowledge Engineering Tool by Intelli Corp., RuleMaster by Radian Corp. In addition, specialized new hardware was developed to run the software with greater speed than before. Companies like Symbolic and LMI introduced computers referred to as LISP machines because they were designed for LISP.

# Architecture

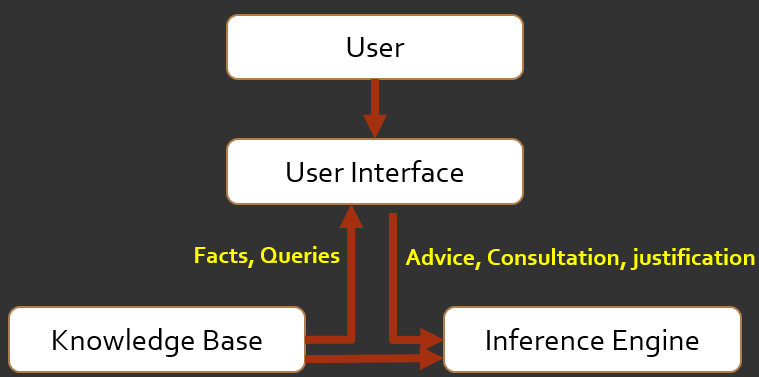
The architecture of expert system reflects the knowledge engineers’ understanding of the methods of representing knowledge and of how to perform intelligent decision making tasks with the support of computer based systems.

Fig: Architecture of an Expert system

The software architecture is independent of the specific computer hardware. The user interface allows users to enter rules and facts about a particular situation and ask questions of the system, provides response to the user request and support all other communication between the system and the users.

The knowledge of the human expert about particular domain is contained in codified form within the Knowledge Base. Even though knowledge bases are developed by experts, one with the knowledge about subject matter can understand the information contained.

Inference engine, uses the information provided to it by the Knowledge Base and the user to infer new facts. This procedure can simulate the deductive thought processes of an expert.

The architecture can be expanded by extending the Knowledge Base into a knowledge database and a domain database. These database can be managed by Database Management System.

Knowledge Database contains rules about the behavior of the elements of the particular subject. The expert system that represent knowledge in rule is known as Rule-Based-Expert-Systems.

Domain Database contains fact about the expert system subjects.

The knowledge information about most expert system are updated and expanded over lifetime of the system. The updated knowledge provides the most relevant and complete assistance to the user. Knowledge acquisition is done by domain expert to ease the transfer of knowledge from human expert to computer based expert system. After acquiring requisite knowledge, rules are put in Knowledge Database and facts in Domain Database

Self-training is another goal of expert system. When an expert system drives new facts through its inference procedure, this new fact could be added to knowledge base as well as given to the user. The self-training facility accepts the fact developed by the expert system’s inference engine and compares these derived facts with the facts stored in the domain database. Is the derived fact is not present in the database then it is added to the database, this is how knowledge if upgraded.

# Goals

The major objective of developing expert system is substituting an unavailable human experts, combining the knowledge and experiences of several human experts, training new experts, providing requisite expertise on project that do not attract or retain experts and providing expertise to the project that can’t afford human expert.

Some organisations may have valuable human expert but there must be substitution in case of unavailability of that expert. Combining the knowledge and experience of several experts is not an easy task, especially if the experts are from more than one subject domain. Expert system with a large amount knowledge, having more information than human experts, can develop inferences that would not be available to human experts.

Human experts needs training and experience. In a subject area in which an organization requires more useful experts, expert systems could prove to be valuable intelligent instruction systems. With an existing expert system in a subject domain, another expert may present the expert system with a problem and request to get a solution. The expert system provides the solution. If the user understand the solution, the user could present the new problem to the system. However, if the user does not understand the solution, he/she may request the system to explain the line of reasoning used to achieve the solution.

There are some activities that are usually not done by experts. System development experts typically leave system development project as they are finishing. These experts do not stay on a project during operation phase. ES could be developed to include the expert’s knowledge about the construction of the particular system. This knowledge will take into account the internal system design, the data flow and the specific system dependencies of a particular system are not always understood by users. Documentations for large systems tend to be extensive and hence challenging for anyone to grasp. ES developed with knowledge about large systems could advise operations and maintenance personnel on operational and maintenance actions. An operator who is trying to get a system to perform a particular action, and cannot figure out how to, query the expert system.

Software maintenance of a large computer system often involves the modification and extension of the software. An ES to assist the maintenance team could be built by the development team during the original construction of the system. This expert system could then advice the maintenance team during maintenance actions so that fixing one system problem does not introduce new problem. It also could update KB with system modification data after each maintenance action. Thus, the expert system stays current with the system it is providing advice for. Organisational priorities and goals sometimes require project to function without the benefit of certain expertse.

# Activities

ES are built to solve many different types of problems. According to their basic activities they are grouped as follows:

|  |  |
| --- | --- |
| **Category** | **Problem Addressed** |
| **Interpretation** | Inferring situation description from sensor data |
| **Prediction** | Inferring the likely consequences of given situation |
| **Diagnosis** | Inferring system malfunction from observations |
| **Design** | Configuring object under constraints |
| **Planning** | Designing actions |
| **Monitoring** | Comparing observations to expected outcomes |
| **Debugging** | Prescribing remedies doe malfunctions |
| **Repair** | Executing plans for administer prescribed remedies |
| **Instruction** | Diagnosing, Debugging and Repairing |
| **Control** | Governing overall system behaviour |

ES that performs interpretation typically use sensor data to infer the situation descriptions. Interpretation systems deal directly with real data rather than with clean symbolic representations of the problem situation. They may face difficulties that may other types of system avoid, because they may have to handle data that are noisy, sparse, incomplete, unreliable or erroneous. They need special technique for extracting data and representing them in symbolically.

# Advantages

ES have many features and some of the advantages of using ES are follows:

* **Increased Availability**: Expertise is available on any suitable computer hardware. In a very real sense, ES is a mass production of expertise.
* **Reduced Cost**: The cost of providing expertise per user is greatly lowered.
* **Reduced Danger**: ES can be used in environment that might be hazardous for human.
* **Permanence**: Unlike human expert, ES won’t retire, quit or die i.e. knowledge will last indefinitely.
* **Multiple expertise:** The knowledge of multiple expert can be made available to work simultaneously and continuously on a problem at any time of day or night. The combined level of expertise of several experts may exceed that of a single human expert.
* **Increased Reliability:** ES can reinforce the confidence in decision made by human experts, by providing a second opinion or break a tie in case of disagreement among multiple human experts. The expert system should always agree with human experts unless a mistake was made by human which may happen when he/she was tired or under stress.
* **Explanation:** ES can explicitly explain the reasoning that led to the conclusion. A human expert may be too tired, unwilling or unable to do this all the time. The ability to reason increases one’s confidence that the correct decision was made.
* **Fast Response**: Fast and real response may be necessary for some applications. Depending on the software and hardware used, an ES may respond faster and be more available than a human expert. Some emergencies may require faster response than a human expert’s, and so a real-time expert system us a good choice.
* **Steady, Unemotional and complete response all the time**: This may be very important in real-time and emergency situations when a human expert may not operate at peak efficiency because of stress or fatigue.
* **Intelligent Tutor**: An EX may act as an intelligent tutor by letting the student run sample programmes and explaining the system’s reasoning.
* **Intelligent Database**: EX can be used to access database intelligently.

# Stages in Development of ES

ES design and development must be carefully programmed if success if desired:

***Outline Statement:*** This stage involves the identification of an appropriate system. The expert and the knowledge engineer work out the concepts, boundaries, relationship, and control mechanisms to be included in the system. Development strategies and constraints as well as user expectations are explored. The results of this stage of assessing the potential performance and benefits of the system should be encapsulated in an outline specification. This specification holds foe the initial prototype development.

***Knowledge acquisition:*** in this stage, there is intensive interaction between the expert and the knowledge engineer. The expert is specific in articulating his knowledge in line with the constraints imposed and tries to highlight the essential issues that set the information apart as being knowledge. The knowledge engineer in the other hand tries comprehend the essence of the knowledge, its limits, and its complexities.

***Knowledge representation:*** once the knowledge engineer has a clear idea about the system, its content, and its limits, he faces the task of designing a method for appropriate knowledge representation. This is a very important part of the work. A successful system will meet the user’s requirements only if the knowledge of the expert is conveyed in a manner understandable to users and in a mode equivalent to what is normally used by human expert to solve a problem. It is the knowledge engineers ability to enter into the frame of reference of both the expert and the potential users and to suitably structure the acquired knowledge that sets him apart as a good or bad engineer.

***Prototype Development:*** after the expert and the knowledge engineer have established the guidelines regarding the functions of the system, the prototyping stage begins. Most system developments proceeds by developing a prototype model prior to establishing a detailed system specification in anticipation of a full blown system. The advantages of building prototype system are:

1. The expert and the knowledge engineer can establish whether system is feasible or not
2. The user can get opportunity to test out the system and to see whether it is likely to meet their requirements.
3. Provides expert and knowledge engineer to with an opportunity to evaluate the cost and the performance of the chosen system.

A skilled knowledge engineer and a motivated and experienced expert can built a prototype system within a limited number of trials.

***Testing:*** It is the process of evaluating the performance and utility of the prototype programme and revising it as necessary. Typically, domain engineer evaluates the prototype and helps knowledge engineer to revise it. As soon as the prototype runs in few examples, testing is scaled up to evaluate to evaluate the performance and utility. Evaluation helps to uncover: missing concepts and relations in representation scheme, knowledge represented at the wrong level of detail, or unwieldy control mechanism. Such problems may force developers to recycle the various development phases, reformulating the concepts, refining the inference rules, and re-testing control flow. The ES must be refined and tested in in a laboratory environment before it can be released for field testing. However, when it is tested by the user on real problems, there may still be some problems in the system and it might take some time to correct it. Users obviously needs high quality performance from the system. They need fast, reliable system which is easy to use and understand and forgives then when they make mistakes. Therefore, ES needs field testing before it is used by commercial users.

***Main Knowledge Acquisition:*** once the prototype is tested and reviewed by the knowledge engineer and the expert, procedures are normally established for the development of the actual system. The first step here is again to access the extent of the knowledge that is required in order to meet the user’s needs. During this stage, it may be decided to involve multiple experts in the acquisition process. The experts check whether the system can or should be integrated with the other system existent within the organization. The overlap of acquired knowledge with other areas within the organization might need to be more fully explored along with the other interfaces, both manual and machine.

***Specification with detailed information:*** the knowledge engineer and the expert provide a detailed system specification during this phase. Some aspects from the ES will face certain correction from first principles, while others will be developed as the consequences of the lesson learned during the prototyping phase. The detailed specification covers the objective of the expanded system, the resource required, the projected time required for implementation, planned costs, system testing and implementation planning.

***System Development:*** this step uses up a sizeable portion of the entire project time allocated. During this stage, it is very important for the user to know exactly how the system is processing, any problems encountered, and the evidence of the new limitations and the new opportunities. Their support during this phase is very useful to the expert and the knowledge engineer. Since this phase requires greater investment of cost and time, it requires careful monitoring.

***Implementation:*** in this phase, ES do not differ significantly from other software system. Implementation procedure should be carried out by the user and supported by the experts. The implementation plan should have been documented during the ‘specification with detailed information’ stage.

***Maintenance:*** an ES need to be maintained as a live system. It requires continuous revision and updating to ensure that the knowledge it contained is always up to date and in accordance with the changing environment in which organizations operate. Maintenance procedures should be formally documented and the responsibility of them out should be assigned in advance.

# Probability based Expert System

ES should incorporate uncertainty since they represent an intuitive decision making process. All intuitive decisions are subject to some uncertainty. It is also reasonable to treat some expert system as decision-making procedures.

To include uncertainty we have to know how to represent rules or statements using probability. The probability of an event A occurring or being *true* can be represented as P(A) or p[A]. The AND combined probability of events A1, A2… A3 can be written as

P(A1 AND A2 AND A3) = P(A1 A2 A3)

The OR combination will have the form

P(A1 OR A2 OR A3) = P(A1 +A2 +A3)

In an ES uncertainty may exist in two possible ways:

***Type-1 uncertainty:*** this exists when there is uncertainty about the truth of the initial inputs, or at least one of them. This uncertainty is then propagated through the rules to final conclusion using some type of probability law for combined events.

***Type-2 uncertainty:*** this exists where is is uncertainty about the validity of the rules in all circumstances, even if the input events are certainly true. This shows that some additional evidence is required, and is missing from the input to the rule.

Probability theory is concerned with uncertain events. For example, we cannot predict in advance whether there will be a sudden change in climatic condition. Such unpredictable events can be called random events. The most general definition of probability is a measure of the uncertainty about the likelihood of an event occurring. We express the probability of an event as the expected relative frequency with which it will occur in the future. The classical definition of probability is that it is the limit of the ratio of the number of observed occurrence of an event to the total number of trials, as the number of trials increases without limits. A trial is the procedure observing the possible occurrence of an event in order to determine whether or not it actually occur. Probability if normally a subjective judgment by an individual of the likelihood of an event occurring. It is a measure of degree of a person’s belief that an event will occur. This is called Subjective probability.

* 1. **Subjective Probability and probability Laws:**In this approach, we emphasize that the probabilities associated with the inputs to an ES should be subjective. The user providing the input is not hampered by the need to cling to relative frequencies at the source of these estimates, but can utilize his her judgment to codify risk estimates. The user of the ES is best qualified to make type-1 risk judgments, since he is making the observations and reporting in them to the system.

On the other hand, for type-2 uncertainties related to the probability that a rule will fire even I all the input arcs are true; it is system experts who can and should make the risk estimates.

Let C denotes the event of an intermediate or final conclusion associated with a node being true, *Ei* the event of the ith lower level condition associated with a node being true.

*E* = event of combined *Ei’s* being true

= event of combined *Ei’s not* being true

= probability of rules firing of combined conditions of the rules are assigned true

= probability of rules firing of combined conditions of the rules are not assigned true

P(E) and P(Ei) are probability that the combined conditions are true and ith condition is true respectively.

is probability that the conclusion at the node is true and the combined condition is true

is probability that the conclusion at node id true and the combined conditions are not true.

P[C] = probability that the rule fires

P[C] = +

P[C] =

* 1. **Monte Carlo Simulation**

Monte Carlo simulation is the most general procedure for solving any combined probability problems.

It can be useful for solving systems containing inputs that are conditional probabilities.

The algorithm is quiet simple:

A sequence of trial that simulate a real life case is executed. The probability of input occurring is denoted by . Some appropriate assumed probabilities are given as inputs. The algorithm begins, for a trial, with the inputs, each one simulated by generating a random number uniformly distributed between 0 and 1.

If the random number is less than , it is assumed that the input is true for current trial.

All nodes are then systematically processed, in an order defined by control structure.

The definition that of the event that a node will be fired is applied to the node.

*E* will be true for AND node if is true, and for OR node if is true.

We can determine if or not C is true by using two random number. If E is true and first random number is less than pre-assigned value of , or if E is false and the second random number is less than pre-assigned value of , then C is true.

An iteration end with the final conclusion being true.

# Expert System Tools

Most expert system make a sharp distinction between the knowledge base and inference engine. We may expect the inference engine to incorporate a number of utilities that make developing an expert system much easier than it would be if we develop it using programming language. The choice of the development tool often drives much of the development process. Since many products by design prescribe an inherent building process. It is important to ensure that the selected tools fits the intended applications.

* 1. **Nature of Expert System Tools**

ES tools are programming systems that simplify the job of constructing an ES. It ranges from very high-level programming language to low level support facilities. We can divide ES tools into four major categories:

1. Programming Language used for ES
2. Knowledge Engineering Languages
3. System Building Aids
4. Support Facilities

**d**

**c**

**b**

**a**

ES Tools

Problem-oriented

Symbol manipulation

Programming

Explanation

Knowledge Acquisition

Design

General purpose

Skeletal

Fig: the main categories of tools available for ES

1. *Programming languages used for expert system:*

The programming language used for ES application are generally problem-oriented languages or symbol manipulation languages. Problem oriented are designed for particular class of problems and symbol manipulation languages are designed for AI application. EX: LISP has a mechanism for manipulation of symbols in the form of list structures. A list is simply a collection of items enclosed within parentheses. List structures are useful building blocks of representing complex concepts. Programming language like LISP offer great flexibility to ES builder but fails to provide guidance on how to represent knowledge, or mechanisms for accessing the knowledge base.

1. *Knowledge engineering languages*

A knowledge engineering language is a sophisticated tools for developing expert systems. It consists of ES building language integrated into an extensive support environment. It can be categorized into Skeletal and general-purpose system. Skeletal knowledge engineering language is simply a stripped-down ES. This is basically an ES with domain specific knowledge removed, leaving only inference engine and the support facilities. A general-purpose knowledge engineering language can handle many different problem areas and types. It gives more control over data access and search than a skeletal system, but may be difficult to use. General-purpose languages vary a great deal in the extent of their generality and flexibility.

1. *System-building aids*

It consists of programmes that facilitate the acquisition and representation of the domain expert’s knowledge, and design of the expert system under construction. These programmes normally focus on different tasks. It can be divided into two basic groups: Design aids and knowledge acquisition aids. Software like AGE provide design aids. This software help knowledge engineer design and built an ES. It provides the user with a set of components which, like building blocks, can be assembled to form portion of ES. Similarly, TEIRESIAS is a nice knowledge acquisition aid. This system building aids facilitate knowledge transfer from domain expert to knowledge base.

1. *Support facilities*

It consists tools for helping with programming, such as debugging aids and knowledge base editor, and tools that enhances the capabilities of the finished system, such as built-in input/output and explanation mechanisms. These facilities are usually available as part of knowledge-engineering languages and are designed to work specially with that language. These are components of ES supporting environment

1. *Debugging mode*

Most programming and knowledge engineering languages contain tracing facilities and break packages. Tracing provides the users with a trace or display of system operations, usually by listing the names of all rules fires or showing the names of subroutines called.

Break packages let the user tall the programme when to stop, so that user can stop the programme execution just before some recurring error and examine the current value in the database.

1. *I/O facilities*

Different ES tools deal with input/output in different ways. These tools, when functioning, acquire knowledge in such a way that the user will be able to communicate simultaneously with the running ES. Expert system tools may handle I/O by providing a set of powerful commands or procedures that make writing I/O routines easy.

1. *Explanation facilities*

ES can explain users how they reach particular conclusion, but not all provide the same degree of software support for explanation. The most common type of explanation mechanism deal with respective reasoning, it explains how the system reached a particular state. For example, the user may want to know how the system responds to a particular question and how it attains its conclusion. Here the system may describe the rule that lead to the question, or display the part of the chain or sequence of rules that lead to the conclusion.

* 1. **Expert System Building Tools**

According to representation technique available in a tool, ES tools can be divided into following types:

1. *Inductive tool*

It generate rules from examples. Normally large inductive tools run in mainframe computers and small inductive tools runs in PCs. These tools are derived from experiments conducted in machine learning. With these tools, a developer feeds large number of examples for the machine information base. The tools use an algorithm to convert the examples into rules and determine the order the system will follow when questioning the user.

1. *Simple and Structured rule base tools*

It uses IF-THEN rules to represent knowledge. This tool is best fitted for developing ES with less than 500 rules. They are simple in contrast to structured rule base tools. They lack editing features available in structured rule base tools. Structured rule based tools offer context trees, multiple instantiation, confidence factor and more powerful editors. These tools uses IF-THEN rules arranged into sets. These “rule sets’ are separate knowledge base. One set of rule can inherit the information acquired when other rule sets are examined. These systems are desirable when large number of rules are involved, if the rules can be subdivided into sets.

1. *Hybrid tools*

It represents the most complex ES development environments currently available. These tools use object-oriented programming technique to represent elements of every problem. An object can contain facts and IF-THEN rules. This tool is used to build a system that contains 500 to several thousand rules and can include features of several different conditions. They usually facilitate the development of complex graphically oriented interfaces.

1. *Domain specific tools*

These tools are specially designed to be used only to develop ES for a particular domain. This tool can incorporate any of the tools stated above. These tools provide special development and user interface that make it possible to develop an ES in a particular domain considerably faster than the types listed above.

* 1. **Expert System Shells**

It is a development system for designing an Expert system. It provides an efficient and user friendly software environment to the knowledge engineer for building an expert programme. A good shell should have features like good debugging and value checking aids, mechanism for handling uncertainties both from developer and end user.

**12.4 Inference Scheme**

Inference engines in ES are responsible for deciding how the knowledge data in the KB should be used. They are responsible for the control and execution of the reasoning strategies used by ES. Backward chaining and forward chaining are strategies used to specify how rules contained in a knowledge base rule system are to be executed.

EX:

**IF** the weather is rainy

**AND** the distance is >= 100 kilometers

**THEN** transportation is by car (**1**)

**IF** transportation is by car

**THEN** passenger insurance will be considered (**2**)

**IF** passenger insurance is considered

**THEN** transportation insurance cost = Rs 10,000 (**3**)

***Backward Chaining:*** suppose we want to establish the fact that “transportation insurance cost = 10000” assuming we know only that “the weather is rainy” and “the distance is 150 km”. Backward chaining works backward from the conclusion:

Is this fact known? 🡪 No

Can it be obtained from the rule? 🡪 Yes, from rule 3

Which fact needs to be known? 🡪 “Passenger insurance is considered”

Is this fact known? 🡪 No

Can it be obtained from the rule? 🡪 yes, from rule 2

Which fact need to know? 🡪 “Transportation by car”

Is this fact known? 🡪 No

Can it be obtained from the rule? 🡪 yes, from rule 1

Which fact need to be known? 🡪 “The weather is rainy” and

“distance >=100 km”

Are these facts known? 🡪 Yes, “The weather is rainy” and

“distance is 150 km”

Therefore, it is true that “transportation is by car”.

Therefore, it is true that “passengers insurance is considered”.

Therefore, it is true that transportation insurance cost = 10000”.

We started with the fact we wanted to prove and tried to establish all the facts needed to reach the goal. This reasoning method is called backward chaining. Backward chaining is applied when a goal or hypothesis is chosen as the starting point for problem solving. Backward chaining is also known as goal-directed, top-down or consequence driven.

***Forward Chaining:*** this approach goes forward from starting point, via the conclusion generated at each step.

Suppose we want to prove that “transportation insurance cost = 10000” assuming we know only that “the weather is rainy” and “the distance is 150 km”

is the fact known? 🡪 No

Which fact do we know? 🡪 “the weather is rainy.” And

“the distance >= 100 km.”

Which fact follow from it? 🡪 “transportation is by car.” Rule(1)

Is this what we want to prove? 🡪 no

What fact follow from it? 🡪 “passenger insurance will be considered.”

Rule(2)

Is this what we want to prove? 🡪 No

What fact follow from it? 🡪 “transportation insurance cost = 10000.”

Rule(3)

Is this what we want to prove? 🡪 Yes.