AI-Unit-5 - Notes given by lecture

Artificial intelligence (Holy Mary Institute of Technology & Science)

5. EXPERT SYSTEM AND APPLICATIONS

INTRODUCTION

Expert systems (ES) are one of the prominent research domains of AI. It is introduced by the researchers at Stanford University, Computer Science Department.

Definition: "The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise."

Characteristics:

1. Expertise: An ES should exhibit expert performance, have high level of skill, and possess adequate robustness. The high-level expertise and skill of an ES aids in problem solving & makes the system cost effective.
2. Symbolic Reasoning: Knowledge in an ES is represented symbolically which can be easily reformulated & reasoned.
3. Self Knowledge: A system should be able to explain & examine its own reasoning.
4. Learning Capability: A system should learn from its mistakes & mature as it grows. Flexibility provided by the ES helps it grow incrementally.
5. Ability to provide Training: Every ES should be capable of providing training by explaining the reasoning process behind solving a particular problem using relevant knowledge.
6. Predictive Modelling Power: This is one of the important features of ES. The system can act as an information processing model of problem solving. It can explain how new situation led to the change, which helps users to evaluate the effect of new facts & understand their relationship to the solution.

Advantages: The Expert Systems are capable of the following:

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem

Disadvantages: The Expert Systems are incapable of the following:

- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

PHASES IN BUILDING EXPERT SYSTEMS

- Identify Problem Domain
- Design the System
- Develop the Prototype
- Test & Refine the Prototype
- Develop & Complete the Expert System
- Maintain the System

Identify Problem Domain:

- The problem must be suitable for an expert system to solve it.
- Find the experts in task domain for the ES project.
- Establish cost-effectiveness of the system.

Design the System:

- Identify the ES Technology.
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Develop the Prototype:

- From Knowledge Base, the knowledge engineer works to
    - Acquire domain knowledge from the expert.
    - Represent it in the form of If-THEN-ELSE rules.

Test & Refine the Prototype:

- The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance.
- End users test the prototypes of the ES.

Develop & Complete the ES:

- Test & ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems.
- Document the ES project well.
- Train the user to use ES.

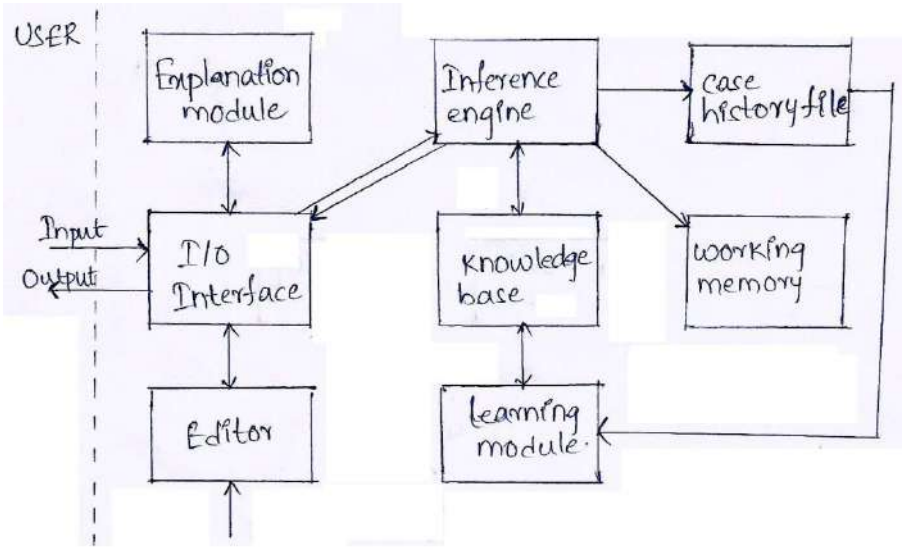
Maintain the System:

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.

EXPERT SYSTEMS VERSUS TRADITIONAL SYSTEMS

Expert System	Traditional System
The entire problem related expertise is encoded in data structures only, not in programs.	Problem expertise is encoded in both program and data structures.
The use of knowledge is vital.	Data is used more efficiently than knowledge.
These are capable of explaining how a particular conclusion is reached and why requested information is needed during a process.	These are not capable of explaining a particular conclusion for a problem. These systems try to solve in a straight forward manner.
Problems are solved more efficiently.	Not as efficient as an Expert System.
It uses the symbolic representations for knowledge i.e. the rules, different forms of networks, frames, scripts etc. and performs their inference through symbolic computations	These are unable to express in symbols. They just simplify the problems in a straight forward manner and are incapable to express the "how, why" questions.
Problem solving tools are present in Expert System	No problem solving tools in specific.
Solution of the problem is more accurate.	Solution of the problem may not be more accurate.
Provide a clear separation of knowledge from its processing.	Do not separate knowledge from the control structure to process this knowledge.
Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a narrow domain.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
Permit inexact reasoning and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, changes are easy to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.

ARCHITECTURE OF EXPERT SYSTEM (or) COMPONENTS OF EXPERT SYSTEM (or) RULE BASED EXPERT SYSTEMS



I/O Interface:

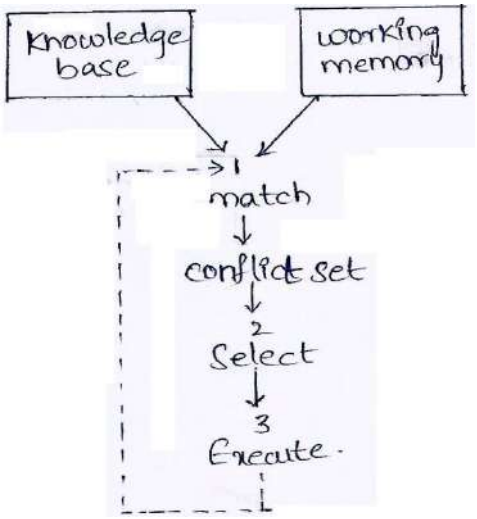
The I/O Interface permits the user to communicate with the system in a simple way.

Knowledge Base:

- The Knowledge Base is one of the important components of Expert System.
- The Knowledge Base consists of a set of production rules & facts (already proven results) along with its solutions.
- For Example, if a student fails to get ≥66% in attendance then declare that student as detained.

Inference Engine:

- Inference Engine is also called as Rule Interpreter.
- It performs the task of matching, from the responses given by the user & the rules.
- Finally it picks up a suitable rule.
- The process is as follows



Explanation Module:

- Explanation Module consists of the Answers for the Questions of 'How' & 'Why'.
- To respond to a 'How' query, the Explanation Module traces the chain of rules.
- To respond to a 'Why' query, the Explanation Module must be able to explain why certain information is needed to complete a step.

Working Memory:

- These are the one which are handled at the current situation by the system.
- It is also called as Temporary Storage Area.

Case History File:

- Case History File consists of all the Input & Output transactions.
- This consists of all the inputs given by various users & the outputs given by the system based upon the input.

Learning Module:

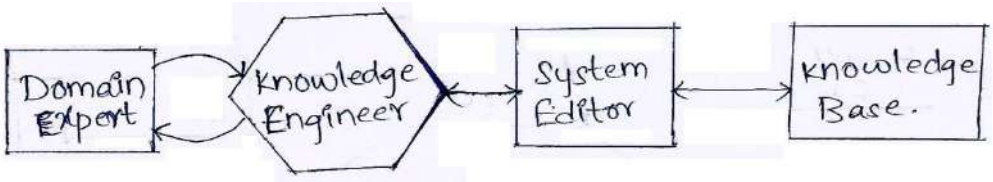
When the data is concerned with Case History File then, the system performs the learning Module.

Editor:

Editor module is used to insert (or) delete the data in the knowledge base.

Knowledge Acquisition & Validation:

One of the most difficult tasks in building knowledge base is in the acquisition & encoding of requisite domain knowledge.



Expert System Shell:

Each Expert System that was build from scratch was done by LISP programming language. After several systems have built in this way it is clear that these systems often use a lot of common things. It was possible to separate the interpreter from the domain specific knowledge & thus to create a system that would be used to construct new Expert Systems by adding new knowledge corresponding to the new problem domain.

One of the important features that a shell must provide is an easy way to interface between an Expert System & a programming environment.

Example: Graphical games.

MYCIN Expert System:

MYCIN is one of the oldest Expert Systems. It was developed at Stanford University in 1970s. It was developed as an ES that could diagnose & recommend treatment for certain blood infections. MYCIN was

developed for exploring the ways in which human experts make guesses on the basis of partial information. In MYCIN, the knowledge is represented as a set of if-then rules. In MYCIN rule can be written in English as follows

- IF the infection is primary-bacteraemia AND the site of the culture is one of the sterile sites AND the suspected portal of entry is the gastrointestinal tract, THEN there is suggestive evidence (0.7) that infection is bacteriod

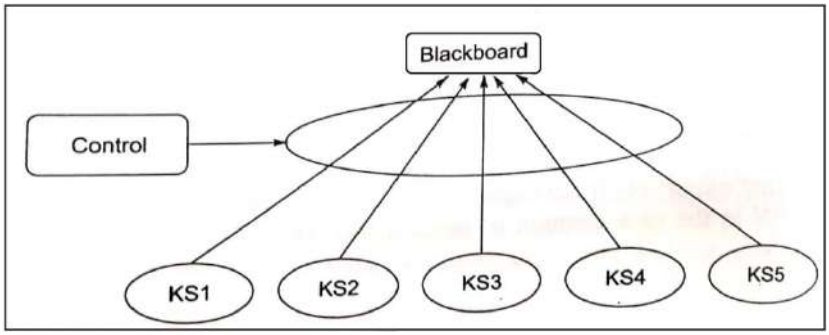
The 3 stages of conversation with MYCIN:

- In First stage, initial data regarding the case was gathered to enable the system to come up with a very broad diagnosis.
- In Second stage, the questions that were asked to test specific hypotheses were more direct in nature.
- In Final stage, a diagnosis was proposed.

The First version of MYCIN had a number of problems which were remedied in later. One of them was that the rules of domain knowledge were often mixed. EMYCIN was the first Expert shell developed from MYCIN. A new ES called PUFF was developed using EMYCIN in the new domain of heart disorders. To train Doctors, the system called NEOMYCIN was developed, which takes them through various sample cases, check their diagnoses, determines whether their conclusions are right & explains where they went wrong.

BLACK BOARD SYSTEMS

Blackboard Systems were first developed in the 1970s to solve complex, difficult & ill-structured problems in a wide range of application areas. Blackboard architecture is a way of representing & controlling the knowledge bases; using independent groups of rules called Knowledge Sources (KSs) that communicate through a data control database called a Blackboard.



The main modules of Blackboard System are as follows. It consists of 3 main components:

- Knowledge Sources
- Blackboard
- Control Shell

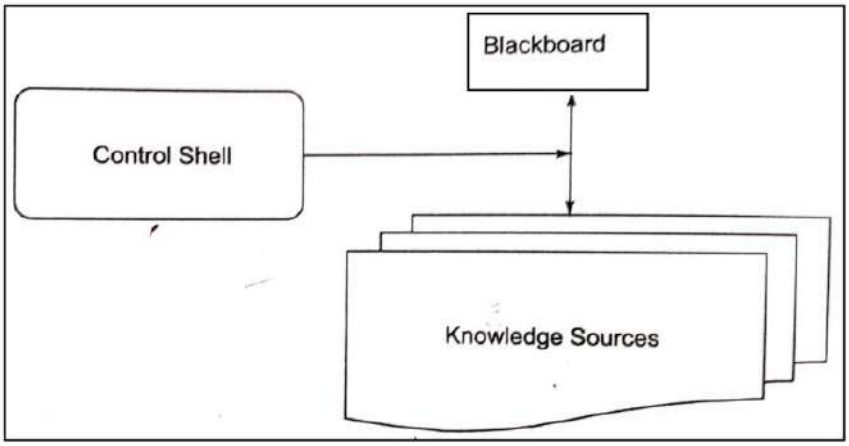


Fig: Blackboard System

Knowledge Sources:

- A KS is regarded to be a specialist at solving certain aspects of the overall application and is separate and independent of all other KSs in the blackboard system.
- Once it obtains the information required by it on the Blackboard, it can proceed further without any assistance from other KSs.
- It is possible to add additional KSs to the Blackboard system and upgrade or even remove existing KSs.
- Each KS is aware of the conditions under which it can contribute toward solving a particular problem. This knowledge in problem solving is known as triggering condition.

Blackboard:

- Blackboard represents a global data repository & shared data structure available to all KSs.
- It contains several important constituents such as raw input data, partial solutions, alternatives & final solutions, control information, communication medium used in various phrases in problem solving.
- An advantage of the blackboard system is that the system can retain the results of problems that have been solved earlier, thus avoiding the task of re-computing them later.

Control Component:

- The control Shell, directs the problem-solving process by allowing KSs to respond to changes made to the black board.
- In a classical blackboard system control approach, the currently executing KS Activation (KSA) generates events as it makes changes on the blackboard.
- These events are ranked & maintained until the executing KSA is completed.

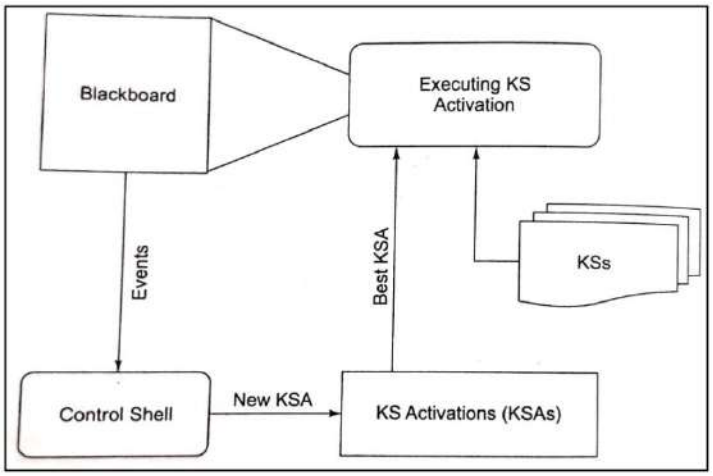


Fig: Blackboard System Control Cycle

Issues in Blackboard Systems for Problem Solving:

- Specialized representation: KSs may only operate on a few classes of blackboard objects.
- Fully general representation: All aspects of blackboard data are understood by all KSs.

Blackboard System versus Rule-based System:

A major difference between a Blackboard System and a Rule-based System lies in the size & scope of rules when compared to the size & complexity of KSs.

TRUTH MAINTENANCE SYSTEMS

Truth Maintenance System (TMS) is a structure which helps in revising set of beliefs & maintaining the truth every time new information contradicts information already present in the system. It is developed by Doyle in 1979. TMS maintains the beliefs for general problem-solving systems. All TMS manipulate proposition symbols & the relationships between different proposition symbols.

- A Monotonic TMS manipulates propositional symbols & Boolean constraints.
- A Non-Monotonic TMS allows for heuristic (or) non-monotonic relationships between proposition symbols such as ‘whenever P is true Q is likely’ (or) ‘if P is true then unless there is evidence to the contrary Q is assumed to be true’.

Monotonic System & Logic:

In Monotonic Systems, once a fact (or) piece of knowledge stored in the knowledge base is identified, it cannot be changed during the process of reasoning. In other words, axioms are not allowed to change as once a fact is confirmed to be true, it must always remain true & can never be modified.

- If a formula is a theorem for a particular formal theory, then that formula remains a theorem for any augmented theory obtained by adding axioms to the theory.

For instance, if a property P is a theorem of T & if T is augmented to T_1 by additional axioms, then P remains a theorem of T_1. Further, if an axiom A is added to a theory T to build a theory T_1, then all the theorems of T are also theorems of T_1.

In Monotonic Reasoning, the world of axioms continually increases in size & keeps on expanding. An example of monotonic form of reasoning is predicate logic.

Non-Monotonic System & Logic:

In non-monotonic systems, truths that are present in the system can be retracted whenever contradictions arise. Hence, the number of axioms can increase as well as decrease. The system is continually updated depending upon the changes in knowledge base. In non-monotonic logic, if a formula is a theorem for a formal theory, then it need not be theorem for an augmented theory. Common sense reasoning is an example of non-monotonic reasoning.

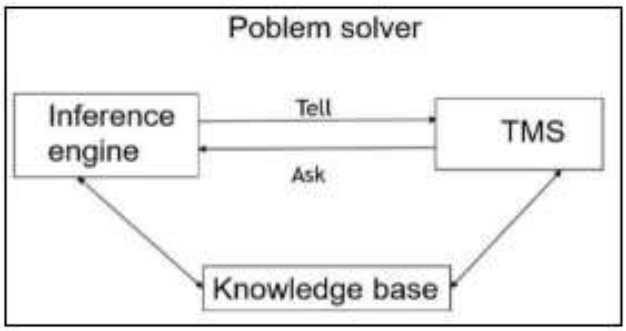


Fig: Interaction between different components in Problem Solver

- The term truth maintenance is synonymous with the term knowledge base maintenance and is defined as keeping track of the interrelations between assertions in knowledge base.
- The main job of TMS is to maintain consistency of the knowledge being used by problem solvers.
- The Inference Engine (IE) solves domain problems based on its current belief set.

Monotonic TMS:

A Monotonic TMs is a general facility for manipulating Boolean constraints on proposition symbols. The constraint has the form P → Q where P & Q are proposition symbols.

Functionality of Monotonic TMS:

- Add_constraint
- Follow_from
- Interface functions

Add_constraint: This interface functions adds a constraint to the internal constraint set. Once a constraint has been added, it can never be removed.

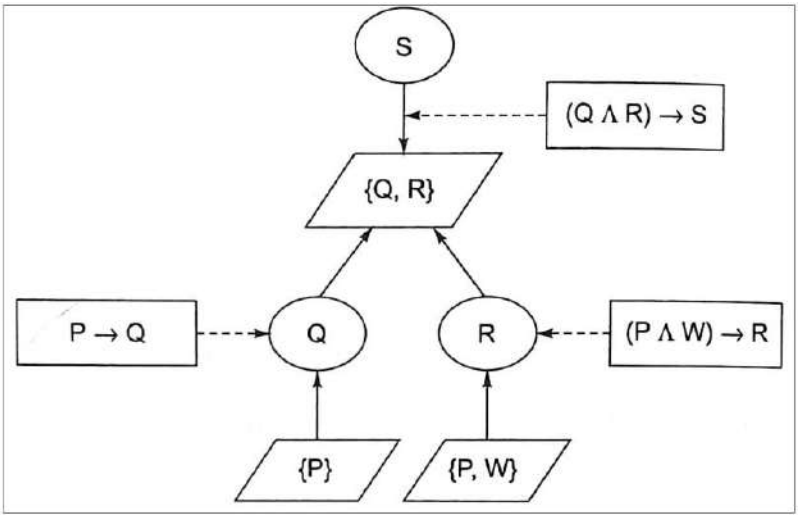
Follow_from: This function takes 2 arguments namely, a literal (L) and a premise set (Σ) and returns the values yes, no or unknown. If yes is returned, then the TMS guarantees that L follows from the Σ and internal constraints. If no is returned, then the TMS guarantees that L does not follows from the Σ and internal constraints. If TMS is unable to determine, then it returns unknown.

Interface Functions: these compute justifications. There are 2 interface functions used to generate such proofs namely justifying literals & justifying constraints.

Let us consider an example with a premise set Σ = {P, W} and an internal constraint set {P → Q, (P Λ W) → R, (Q Λ R) → S}.

Justification Literals	Derived Literals	Justifying Constraints
{P, W}	R	(P Λ W) → R
{P}	Q	P → Q
{Q, R}	S	(Q Λ R) → S

The Justification Tree is as follows



Non-monotonic TMS:

The basic operation of a TMS is to attach a justification to a fact. A fact can be linked with any component of program knowledge which is to be connected with other components of program information.

Support List is defined as SL (IN-node) (OUT-node), where IN-node represents a list of all IN-nodes (propositions) that support the considered node as true.

- Here, IN means that the belief is true. OUT-node is a list of all OUT-nodes that do not support the considered node as true.
- OUT means that the belief is not true for considered node.

For example

Node Number	Facts/assertions	Justification (justified belief)
1	It is sunny	SL(3) (2,4)
2	It rains	SL() ()
3	It is warm	SL(1) (2)
4	It is night time	SL() (1)

Table: Justification of Facts

In this case

- An empty list indicates that its justification does not depend on the current belief or disbelief.
- Node 1 assumes that it is sunny, provided that it is warm and it does not rain, and it is not night time.
- Node 2 has both empty lists indicating that its justification does not depend on current beliefs or disbeliefs.
- Node 3 assumes that it is warm given that it is sunny & it does not rain.
- Node 4 does not depend on the list in its (IN-node) part.

APPLICATIONS OF EXPERT SYSTEMS

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Prediction	Perform the task of inferring the likely consequences of a situation like weather prediction for rains, storms, prediction of crops, share market etc.
Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

LIST OF SHELLS & TOOLS

The following are the list of popular shells & tools

- ACQUIRE
- ARITY
- ART
- CLIPS

ACQUIRE: It is primarily a knowledge-acquisition system & an ES shell, which provides a complete development environment for the building & maintenance of knowledge-based applications. ACQUIRE SDK is a software development kit, which provides callable libraries for systems such as MS-DOS, Windows, Windows NT, Windows 95 and Win 32.

ARITY: Expert Development Package is an ES that was developed by ARITY Corporation.

ART: It is called Automated Reasoning Tool that is an ES shell, which is based on LISP. It supports rule-based reasoning, hypothetical reasoning and case-based reasoning.

CLIPS: It is used to represent C Language Integrated Production System. It is a public domain software tool that is used for building Expert Systems. CLIPS is probably the most widely used ES tool because it is fast, efficient and free.

FLEX: Flex is a hybrid ES which is implemented in PROLOG. It supports forward & backward chaining, multiple inheritance & also possesses an automatic question & answer system.

GENSYM'S G2: It offers a graphical, object oriented environment for the creation of intelligent applications that are able to monitor, diagnose & control dynamic events in various environments.

GURU: It is an ES developed environment & offers a wide variety of information processing tools combined with knowledge-based capabilities such as forward chaining, backward chaining, mixed chaining, multi-value variables & fuzzy reasoning.

HUGIN SYSTEM: It is a software package for construction of model-based Expert System. It is easy-to-use.

KNOWLEDGE CRAFT: It is an ES development tool kit for scheduling, design & configuration applications.

K- VISION: It is a knowledge acquisition & visualization tool. It runs on Windows, DOS & UNIX workstations.

MAILBOT: It is a personal e-mail agent that reads an e-mail message on standard input & creates an e-mail reply to be sent to the sender of the original message. It provides filtering, forwarding, notification & automatic question-answering capabilities.

TMYCIN: It is an acronym for Tiny EMYCIN & is an ES shell modeled after the EMYCIN shell that was developed at Stanford. It is especially useful for student exercises, although real ES have been written using it. TMYCIN is written in common LISP & is fairly small.