UNIT –V Expert Sytems

Expert Systems, Phases in building expert systems, Expert System Architecture, Rule Based Systems, Forward Chaining,Blackboard Systems, Blackboard architecture, Blackboard System vs Rulebased system, Truth maintenance system

5.1 Expert Systems (ES):

- Expert systems are knowledge based programs which provide expert quality solutions to the problems in specific domain of applications.
- The core components of expert system are
  - knowledge base and
  - navigational capability (inference engine)
- Generally its knowledge is extracted from human experts in the domain of application by knowledge Engineer.
  - Often based on useful thumb rules and experience rather than absolute certainties.
- A process of gathering knowledge from domain expert and codifying it according to the formalism is called knowledge engineering.

5.2 Phases in building Expert System

- There are different interdependent and overlapping phases in building an expert system as follows:
- ***Identification Phase*:**
  - Knowledge engineer finds out important features of the problem with the help of domain expert (human).
  - He tries to determine the type and scope of the problem, the kind of resources required, goal and objective of the ES.
- ***Conceptualization Phase*:**
  - In this phase, knowledge engineer and domain expert decide the concepts, relations and control mechanism needed to describe a problem solving.

- ***Formalization Phase:***
  - It involves expressing the key concepts and relations in some framework supported by ES building tools.
  - Formalized knowledge consists of data structures, inference rules, control strategies and languages for implementation.
- ***Implementation Phase:***
  - During this phase, formalized knowledge is converted to working computer program initially called prototype of the whole system.
- ***Testing Phase:***
  - It involves evaluating the performance and utility of prototype systems and revising it if need be. Domain expert evaluates the prototype system and his feedback help knowledge engineer to revise it.

5.3. Expert System Architecture

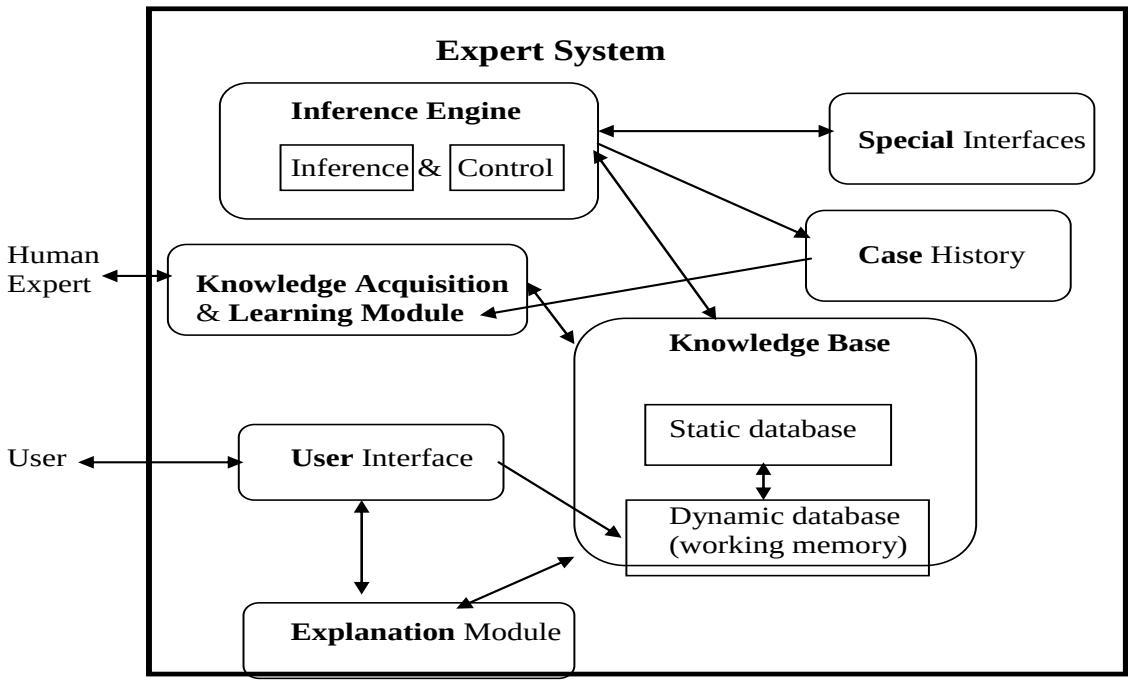


Fig: Architecture of Expert System

Knowledge Base (KB)

- KB consists of knowledge about problem domain in the form of static and dynamic databases.
- Static knowledge consists of
  - rules and facts which is complied as a part of the system and does not change during execution of the system.
- Dynamic knowledge consists of facts related to a particular consultation of the system.
  - At the beginning of the consultation, the dynamic knowledge base often called working memory is empty.
  - As a consultation progresses, dynamic knowledge base grows and is used along with static knowledge in decision making.

Working memory is deleted at the end of consultation of the system

Inference Engine

- It consists of inference mechanism and control strategy.
- Inference means search through knowledge base and derive new knowledge.
- It involve formal reasoning involving matching and unification similar to the one performed by human expert to solve problems in a specific area of knowledge.
- Inference operates by using modus ponen rule.
- Control strategy determines the order in which rules are applied.
- There are mainly two types of control mechanism viz., forward chaining and backward chaining.

Knowledge Acquisition

- Knowledge acquisition module allows system to acquire knowledge about the problem domain.
- Sources of Knowledge for ES

  - text books, reports, case studies,
  - empirical data and
  - domain expert experience.

- Updation of Knowledge can be done using knowledge acquisition module of the system.
  - insertion,
  - deletion and
  - updation of existing knowledge

Case History

- Case History stores the file created by inference engine using the dynamic database created at the time of consultation.
- Useful for learning module to enrich its knowledge base.
- Different cases with solutions are stored in Case Base system.
- These cases are used for solving problem using Case Base Reasoning (CBR).

Explanation module

- Most expert systems have explanation facilities that allow the user to ask the system ***why*** it asked some question, and ***how*** it reached to conclusion.
- It contains 'How' and 'Why' modules attached to it.
  - The sub-module 'How' tells the user about the process through w
  - hich system has reached to a particular solution
  - 'Why' sub-module tells that why is that particular solution offered.
- It explains user about the reasoning behind any particular problem solution.
- Questions are answered by referring to the system goals, the rules being used, and any existing problem data.

Explanation Modules Responses

User	**Why module**
System	Do you have running nose?
User	Yes.
System	I am trying to prove that you have cold. Do you have headache?
User	Yes
System	Do you have sore throat?
User	No
System	Then you do not have cold. Do you have fever?
User	Yes
System	Now I am looking at the possibility of measles. Do you have cough?
User	Yes
System	I can infer measles using rule "If symptoms are fever, cough, running_nose, then patient has measles" measles is concluded.
User	**How Module**
System	Since you have fever, running_nose and cough and there is a rule "If symptoms are fever, cough, running_nose, then patient has measles". So measles is concluded for you.

User Interfaces

- Allows user to communicate with system in interactive mode and helps system to create working knowledge for the problem to be solved.

Dialogue Module (User Interface)	
System	Do you have fever?
User	Yes
System	Do you have bad throat?
User	No
System	Do you have cough?
User	Yes
System	Are you suffering from running nose?
User	Yes
System	Are you suffering from headache?
User	No

Special interfaces

- It may be used for specialized activities such as handling uncertainty in knowledge.
- This is a major area of expert systems research that involves methods for reasoning with uncertain data and uncertain knowledge.

- Knowledge is generally 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*.

5.4 Rule Based Expert Systems

- A rule based expert system is one in which knowledge base is in the form of rules and facts.
  - Knowledge in the form of rules and facts is most popular way in designing expert systems.
- It is also called *production system*.
- Example: Suppose doctor gives a rule for measles as follows:

"**If symptoms are fever, cough, running_nose, rash and conjunctivitis then patient probably has measles".**

- Prolog is most suitable for implementing such systems.

```
hypothesis(measles) :- symptom(fever), symptom(cough),
                       symptom(running_nose),symptom(conjunctivitis),
                       symptom(rash).
```

Simple Medical diagnostic system with dynamic databases:

- The system starts with consultation predicate, that initiates dialog with user to get information about various symptoms.
- Positive and negative symptoms are recorded in dynamic database and '**hypothesis(Disease)**' is satisfied based on stored facts about symptoms.
- If the hypothesis goal is satisfied then the disease is displayed otherwise display 'sorry not able to diagnose'.
- Finally in both the situations, symptom database for a particular user is cleared.

Query:?-consultation.

Medical Consultation System

```
consultation        :- writeln('Welcome to MC System'),
                       writeln('Input your name),
                       readln(Name),

                       hypothesis(Dis), !,

                       writeln(Name, 'probably has', Dis),
                       clear_consult_facts.

consultation        :- writeln('Sorry, not able to diagnose'),
                       clear_consult_facts

hypothesis(flu)     :- symptom(fever),
                       symptom(headache),

                       symptom(body_ache),
                       symptom(sore_throat),
                       symptom(cough),
                       symptom(chills),

                       symptom(running_nose),
                       symptom(conjunctivitis).

symptom(fever)      :- positive_ symp('Do you have
                       fever(y/n) ?', fever).

symptom(cough)      :- positive_ symp('Do you have
                       cough (y/n) ?', cough).

symptom(chills)     :- positive_ symp('Do you have
                       chills (y/n) ?', chills).

positive_ symp(_, X)  :- positive(X), !.

positive_ symp(Q, X)  :- not(negative(X)),
                         query(Q, X, R), R = 'y'.

query(Q, X, R)      :- writeln(Q), readln(R),
                       store(X, R).

store(X, 'y')       :- asserta(positive(X)).

store(X, 'n')       :- asserta(negative(X)).
```

clear_consult_facts :- retractall(positive(_)).

clear_consult_facts :- retractall(negative(_)).

5.5.Forward Chaining

- Prolog uses backward chaining as a control strategy, but forward chaining can be implemented in Prolog.
- In forward chaining, the facts from static and dynamic knowledge bases are taken and are used to test the rules through the process of unification.
- The rule is said to be fired and the conclusion (head of the rule) is added to the dynamic database when a rule succeeds.
- Prolog rules are coded as facts with two arguments, first argument be left side of rule and second is the list of sub goals in the right side of the rule.
- Represent prolog rule as a fact by **rule_fact** predicate and simple facts by **fact** predicate.
- Consider the following Prolog rules and facts with their corresponding new fact representations.

a:-b⇒rule_fact(a, [b]).

c:-b, e, f.⇒rule_fact(c, [b, e, f]).

b.⇒fact(b).

e.⇒fact(e).

f.⇒fact(f).

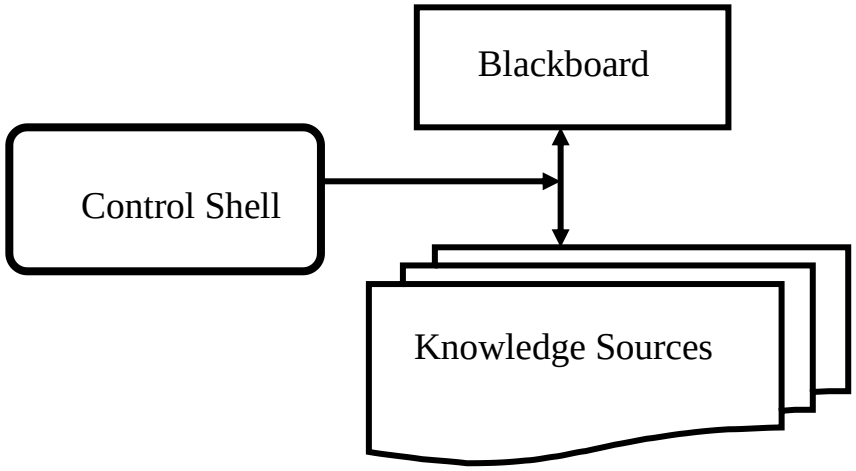
- Here a, b, c, e, f are atoms (predicates with arguments, if any).
- Newly generated facts are stored in database file 'dfile' which is consulted in the prolog program.

5.6.Blackboard System – BS

- Blackboard systems are used to capture dynamic environment with the help of domain experts.

- BS uses a functional modularization of expertise knowledge in the form of Knowledge Sources (KS).
- KS are independent computational modules containing the expert knowledge needed to solve the problem.
  - Blackboard approach has an ability to integrate contributions dynamically for which relationships would be difficult to specify by the KS writer in advance.
- BS consists of three main components viz., Knowledge Sources, Blackboard and Control Shell.
  - BS does not allow direct interaction among modules, as all communication is done via the blackboard through control shell.

5.7.Blackboard System Architecture



Knowledge Source – KS

- KS can be widely diverse in their internal representation and computational techniques and they do not interact directly with each other.
- KS is a specialist at solving certain aspects of the overall application and is separate and independent of all other KSs.
- Once it finds the information it needs on the blackboard, it can proceed without any assistance from other KSs.

- Additional KSs can be added to the blackboard system, existing KS can be upgraded or even can be removed.
- Each KS is aware of its conditions under which it can contribute toward solving the problem.

Blackboard

- The blackboard is a global data repository and shared data structure available to all KSs
- It contains raw input data, partial solutions and final solutions, control information, communication medium etc.
- The system can retain the results of problem-solved earlier, thus avoiding re-computing them later.
- Structuring of information on blackboard is important issue.
- It should enable a KS to efficiently inspect the blackboard to see if relevant information is present.

Control Shell

- The control shell directs the problem-solving process by allowing KSs to respond opportunistically to changes made to the blackboard.
- The control shell reports about the kind of events in which each KS is interested in.
- It maintains this triggering information and directly considers the KS for activation whenever that kind of event occurs.

Information Representation on Blackboard

- There are two ways for storing information on blackboard viz.,
  - specialized representation and
  - fully general representation.
- In specialized representation,
  - KSs may only operate on a few classes of blackboard objects.
  - Sharing data by only a few KSs limits the extendibility and scalability of the system.

- In fully general representation, all aspects of blackboard data are understood by all KSs.

There is a trade-off between these two representations

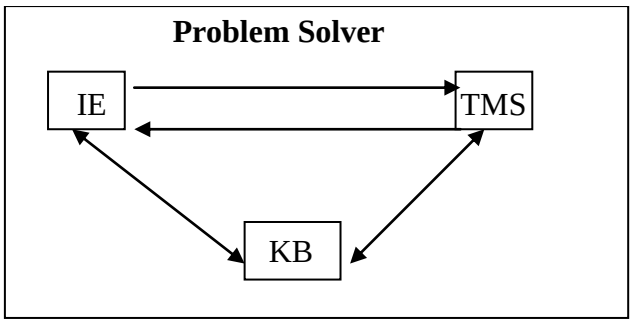
Blackboard System verses Rule Based System

- A blackboard system is different from a rule-based system especially in the size and scope of rules versus the size and complexity of KSs.
- The KSs are substantially larger and more complex than each isomorphic rule in an expert system.
- While expert systems work by firing a rule in response to stimuli, a blackboard system works by executing an entire KS in response to an event.
- A single KS in a blackboard system could be implemented as a complete rule-based system.

5.8.Truth Maintenance System (TMS)

- Truth maintenance system (TMS) works with inference engines for solving problems within large search spaces.
- The TMS and inference engine both put together can solve problems where algorithmic solutions do not exist.

TMS maintains the beliefs for general problem solving systems.



- TMS can be used to implement monotonic or non-monotonic systems.
- In monotonic system, once a fact or piece of knowledge is stored in KB, it can not change.

- In monotonic reasoning, the world of axioms continually increases in size and keeps on expending.
- Predicate logic is an example of monotonic form of reasoning. It is a deductive reasoning system where new facts are derived from the known facts.

- Non-monotonic system allows retraction of truths that are present in the system whenever contradictions arise.
  - So number of axioms can both increase and decrease and depending upon the changes in KB, it can be updated.

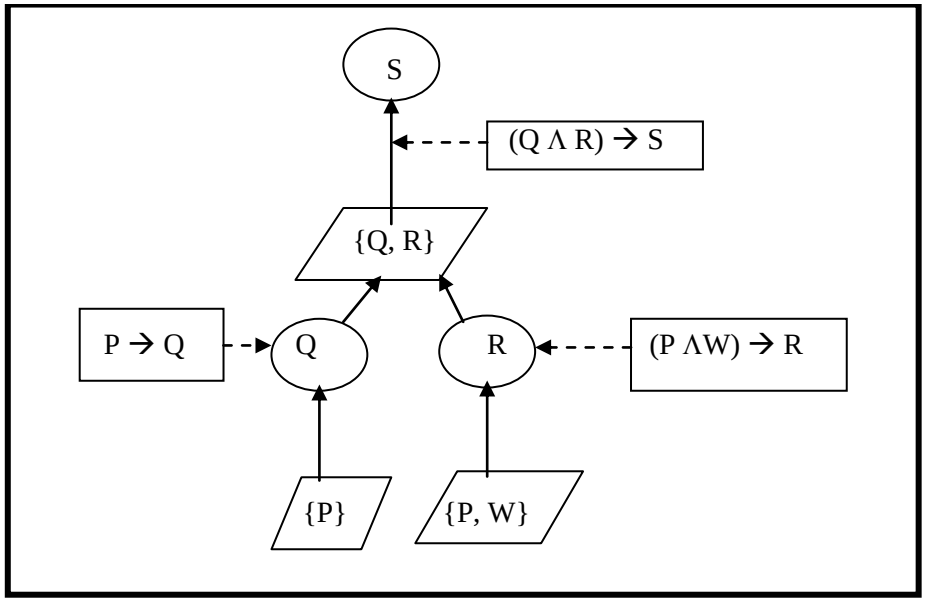
Example – Monotonic TMS

- Suppose we are given the premise set $\sum = \{P, W\}$ and the internal constraint set

$$\{P \rightarrow Q, (P \wedge W) \rightarrow R, (Q \wedge R) \rightarrow S\}.$$

- TMS are able to derive S from these constraints and the premise set $\sum$.
- TMS should provide the justifications of deriving S from constraints and premises.
- Therefore, for any given set of internal constraints and premise set $\sum$, if a formula S can be derived from these, then justification functions generate a justification tree for S.

Justification Tree



Non-Monotonic TMS:

- TMS basically operates with two kinds of objects
  - 'Propositions' declaring different beliefs and
  - 'Justifications' related to individual propositions for backing up the belief or disbelief expressed by the proposition.
- For every TMS, there are two kinds of justifications required namely 'Support list' and 'Conditional proof'.

Support list (SL):

- It is defined as "SL(IN-node)(OUT-node)", where IN-node is 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 for the considered node to be true. OUT means that belief is not true.

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)