2-Extert System

What is Expert System?

- An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert.
- It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.
- The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence.
- It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base.
- The system helps in decision making for complex problems using **both facts and heuristics like a human expert**.
- It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain.
- These systems are designed for a specific domain, such as **medicine, science,** etc.
- The performance of an expert system is based on the expert's knowledge stored in its knowledge base.
- The more knowledge stored in the KB, the more that system improves its performance.
- One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Phases in Building Expert System

The phase in Expert System are:

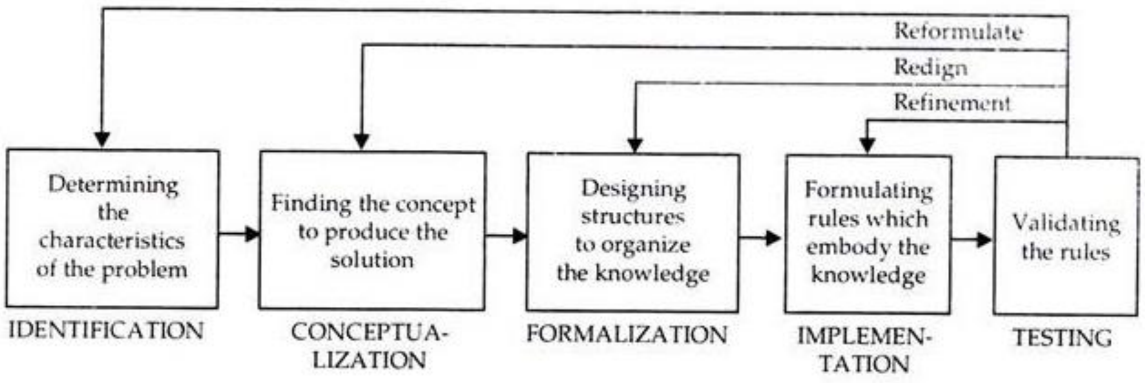
1. Identification

2. Conceptualisation

3. Formalisation (Designing)

4. Implementation

5. Testing (Validation, Verification and Maintenance).


Reformulate
Redign
Refinement
Determining the characteristics of the problem
Finding the concept to produce the solution
Designing structures to organize the knowledge
Formulating rules which embody the knowledge
Validating the rules
IDENTIFICATION
CONCEPTUA-LIZATION
FORMALIZATION
IMPLEMEN-TATION
TESTING

1. Identification:

- Before we can begin to develop an expert system, it is important to describe, with as much precision as possible, the problem which the system is intended to solve.

- It is not enough simply to feel that an expert system would be helpful in a certain situation; we must determine the exact nature of the problem and state the precise goals which indicate exactly how the expert system is expected to contribute to the solution.

Conceptualization:

- In the conceptualization stage, the knowledge engineer frequently creates a diagram of the problem to depict graphically the relationships between the objects and processes in the problem domain.
- It is often helpful at this stage to divide the problem into a series of sub-problems and to diagram both the relationships among the pieces of each sub-problem and the relationships among the various sub-problems.

Formalization (Designing):

- In the preceding stages, no effort has been made to relate the domain problem to the artificial intelligence technology which may solve it.
- During the identification and formalization stages, the focus is entirely on understanding the problem.
- Now, during the formalization stage, the problem is connected to its proposed solution, an expert system is supplied by analyzing the relationships depicted in the conceptualization stage.
- The knowledge engineer begins to select the techniques which are appropriate for developing this particular expert system.

Implementation:

- During the implementation stage the formalized concepts are programmed into the computer which has been chosen for system development, using the predetermined techniques and tools to implement a ‘first-pass’ (prototype) of the expert system.

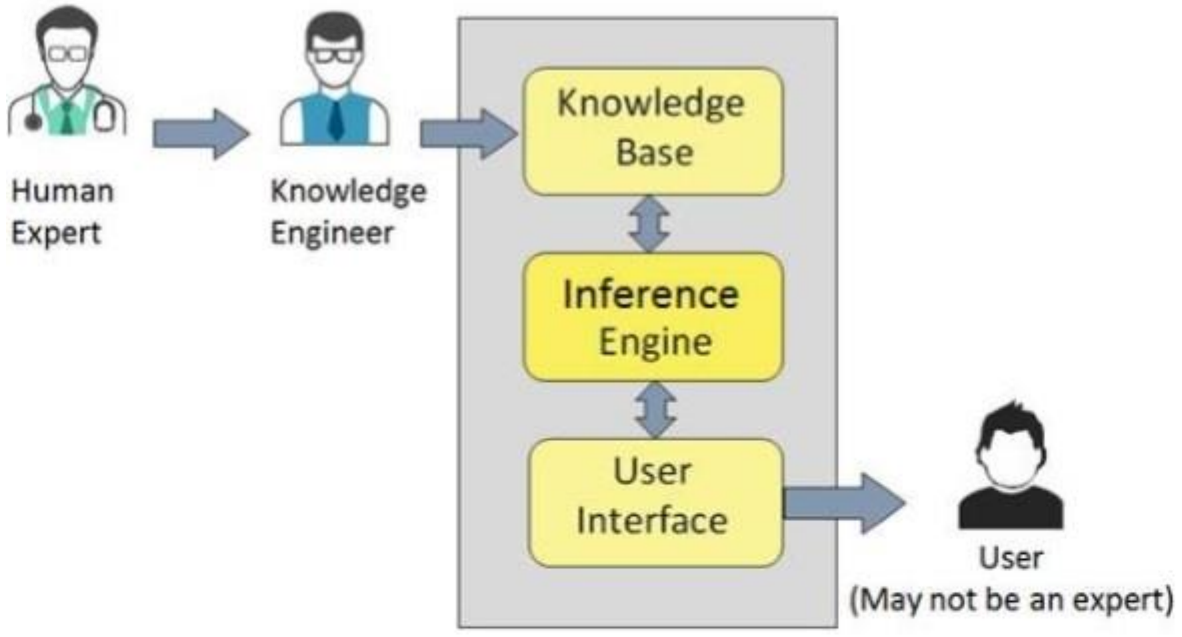
Testing (Validation, Verification and Maintenance):

- The chance of prototype expert system executing flawlessly the first time it is tested are so slim as to be virtually non-existent.
- A knowledge engineer does not expect the testing process to verify that the system has been constructed entirely correctly.
- Rather, testing provides an opportunity to identify the weaknesses in the structure and implementation of the system and to make the appropriate corrections.

Components of Expert Systems

The components of ES include −

- **Knowledge Base**
- **Inference Engine**
- **User Interface**



Knowledge Base

- It contains domain-specific and high-quality knowledge.
- Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

What is Knowledge?

- The data is collection of facts. The information is organized as data and facts about the task domain. **Data, information,** and **past experience** combined together are termed as knowledge.

Knowledge representation

- It is the method used to organize and formalize the knowledge in the knowledge base. It is in the form of IF-THEN-ELSE rules.

Knowledge Acquisition

- The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledge base.
- The knowledge base is formed by readings from various experts, scholars, and the **Knowledge Engineers**.
- The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills.

Inference Engine

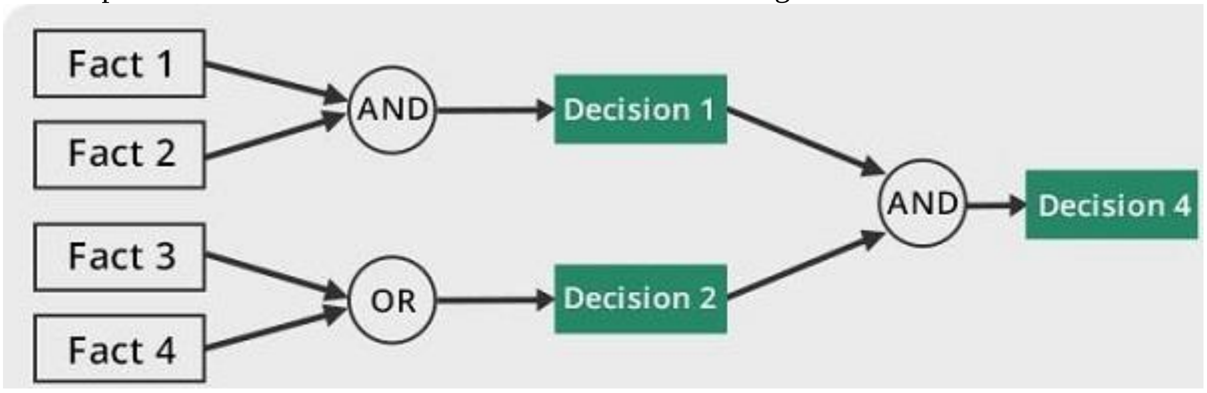
- Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution.
- In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

The Inference Engine uses the following strategies −

- **Forward Chaining**
- **Backward Chaining**

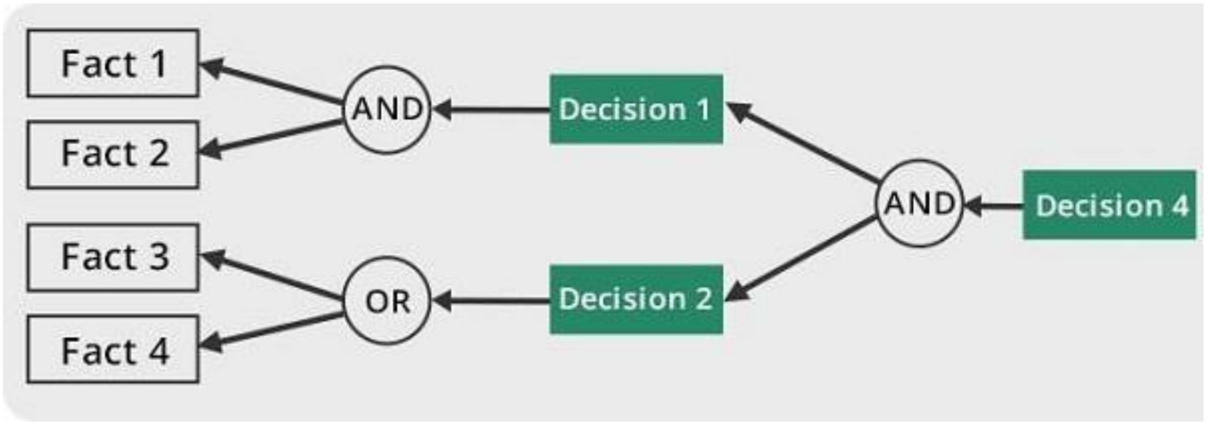
Forward Chaining

- It is a strategy of an expert system to answer the question, **"What can happen next?"**
- Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome.
- It considers all the facts and rules, and sorts them before concluding to a solution.
- This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

- With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**
- On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result.
- This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



User Interface

- User interface provides interaction between user of the ES and the ES itself.
- It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain.
- The user of the ES need not be necessarily an expert in Artificial Intelligence.

Expert Systems Limitations

No technology can offer easy and complete solution. Large systems are costly, require significant development time, and computer resources. ESs have their limitations which include −

- Limitations of the technology
- Difficult knowledge acquisition
- ES are difficult to maintain
- High development costs

Applications of Expert System

The following table shows where ES can be applied.

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.
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.

Expert System Technology

There are several levels of ES technologies available. Expert systems technologies include −

- **Expert System Development Environment** − The ES development environment includes hardware and tools. They are −
  - Workstations, minicomputers, mainframes.
  - High level Symbolic Programming Languages such as **LIS**t **P**rogramming (LISP) and **PRO**grammation en **LOG**ique (PROLOG).
  - Large databases.

- **Tools** − They reduce the effort and cost involved in developing an expert system to large extent.
  - Powerful editors and debugging tools with multi-windows.
  - They provide rapid prototyping
  - Have Inbuilt definitions of model, knowledge representation, and inference design.
- **Shells** − A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility. For example, few shells are given below −
  - Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system.
  - Vidwan, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules.

DIFFERENCE BETWEEN EXPERT SYSTEM AND CONVENTIONAL SYSTEM

- In conventional applications, problem expertise is encoded in both program and data structures. In the expert system approach all of the problem related expertise is encoded in data structures only, none is in programs.

- Generally in expert systems, the use of knowledge is vital. But in conventional system data is used more efficiently than knowledge.

- Conventional systems are not capable of explaining a particular conclusion for a problem. These systems try to solve in a straight forward manner. But expert systems are capable of explaining how a particular conclusion is reached and why requested information is needed during a process. However, the problems are solved more efficiently than a conventional system by an expert system.

- Generally in 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. But conventional systems are unable to express these terms.

- Perform knowledge and decision reasoning tasks vs. performing programmed step-by-step procedures

Rule-based Expert System (also known as production systems or expert systems)

Rule Base System

- If-then rules are one of the most common forms of knowledge representation used in expert systems.
- Systems employing such rules as the major representation paradigm are called rule based systems.

- Some people refer to them as production systems. There are some differences between rule based systems and production systems, but we will ignore these differences and use the terms interchangeably.
- In computer science, a rule-based system is used to store and manipulate knowledge to interpret information in a useful way. It is often used in artificial intelligence applications and research.
- Rule-based systems constructed using automatic rule inference, such as rule-based machine learning.
- One of the first popular computational uses of rule based systems was the work by Newell and Simon on the General Problem Solver [Newell and Simon, 1972].

Applications

- A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices.
- For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms, or select tactical moves to play a game.
- Rule-based systems can be used to perform analysis to compile or interpret computer programs, or in natural language processing.
- Rule-based programming attempts to derive execution instructions from a starting set of data and rules.
- This is a more indirect method than that employed by an imperative programming language, which lists execution steps sequentially.
- Rule Base
- The rule base (also called the knowledge base) is the set of rules which represents the knowledge about the domain. The general form of a rule is:

```
If cond1
and cond2
and cond3
...
then action1, action2, ...
```

The conditions cond1, cond2, cond3, etc((also known as antecedents)).

Some systems would allow disjunctions in the antecedents. For example, rules like the following would be allowed.

```
If cond1 and cond2 or cond3

...

then action1, action2, ...
```

- Such rules are interpreted to mean that if the antecedents of the rule together evaluate to true (i.e., if the Boolean combination of the conditions is true), the actions in the consequents (i.e., action1, action2, etc.) can be executed.
- For example, an antecedent in a rule in a medical expert system could be: the patient has previously undergone heart surgery.
- The complexity of antecedents can vary a lot depending on the type of language used. For instance, in some languages.
- In this chapter we will consider rules with only one consequent and one or more antecedents which are combined with the operator and. We will use a representation of the form:

```
ruleid: If antecedent1 and antecedent2 .... then consequent

For instance, to represent the rule that all birds can fly, we use:
f1: If bird(X) then can_fly(X)
```

- if you want to represent the knowledge that either a bird or 1 a plane can fly, you can do this by using two rules f1 and f2 as follows:

```
f1: If bird(X) then can fly(X)
f2: If plane(X) then can fly(X)
```

- Therefore the disjunction (ORing) of a set of antecedents can be achieved by having different rules with the same consequent.
- Similarly, if multiple consequents follow from the conjunction (ANDing) of a set of antecedents, this knowledge can be expressed in the form of a set of rules with one consequent each. Each rule in this set will have the same set of antecedents.

Advantages of Rule Based Systems

Some of the advantages of rule based systems are:

Because of the uniform syntax, the meaning and interpretation of each rule can be easily analyzed.

1. Homogeneity

- Because of the uniform syntax, the meaning and interpretation of each rule can be easily analyzed.
- Simplicity Since the syntax is simple, it is easy to understand the meaning of rules.
- Domain experts can often understand the rules without an explicit translation.
- Rules therefore can be self-documenting to a good extent.

2. Independence

- While adding new knowledge one need not be worried about where in the rule base the rule is added, or what the interactions with other rules are.
- In theory, each rule is an independent piece of knowledge about the domain.
- However, in practice, this is not completely true, as we shall see in the next section.
- Modularity The independence of rules leads to modularity in the rule base.
- You can create a prototype system fairly quickly by creating a few rules.
- This can be improved by modifying the rules based on performance and adding new rules.

3. Knowledge is separated from Use and Control

- The separation of the rule base from the inference engine separates the knowledge from how it is used to solve the problem.
- This means that the same inference engine can be used with different rule bases and a rule base can be used with different inference engines.
- This is a big advantage over conventional programs where data and control are intermixed.
- Procedural Interpretations Apart from declarative interpretation, rule based systems have procedural interpretations also, which enable them to be viewed as computational models.

Blackboard System

- A blackboard system is an artificial intelligence approach based on the blackboard architectural model, where a common knowledge base, the "blackboard", is iteratively updated by a diverse group of specialist knowledge sources, starting with a problem specification and ending with a solution.
- Each knowledge source updates the blackboard with a partial solution when its internal constraints match the blackboard state.
- In this way, the specialists work together to solve the problem. The blackboard model was originally designed as a way to handle complex problems, where the solution is the sum of its parts.

Blackboard System for Problem Solving

- A blackboard system can be viewed as a group of sitting human specialists next to a large blackboard. They are working cooperatively in order to solve the problem and they use the blackboard as a workplace for solution development.
- Problem solving begins with announcement of a problem and writing initial data onto the blackboard. The specialists are watching the blackboard looking for an opportunity in order to make contribution for solution development.
- When a specialist finds this opportunity, he records the contribution on the blackboard, in hope that others will use his contribution for final problem solving. This process continues until the problem is solved.

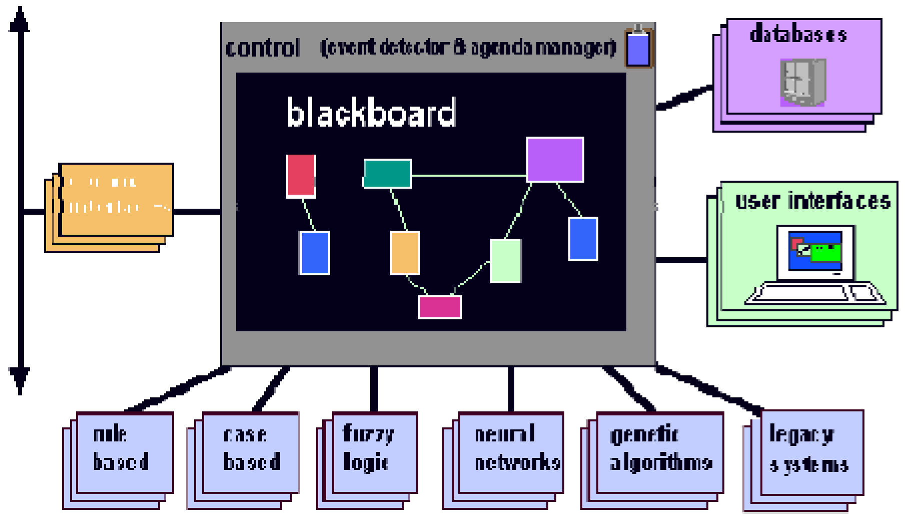
- A group of specialists are seated in a room with a large blackboard. They work as a team to brainstorm a solution to a problem, using the blackboard as the workplace for cooperatively developing the solution.
- The session begins when the problem specifications are written onto the blackboard.
- The specialists all watch the blackboard, looking for an opportunity to apply their expertise to the developing solution.
- When someone writes something on the blackboard that allows another specialist to apply their expertise, the second specialist records their contribution on the blackboard, hopefully enabling other specialists to then apply their expertise.
- This process of adding contributions to the blackboard continues until the problem has been solved.

Components of Blackboard System

- A blackboard system consists of three components:
  1) Knowledge sources (KSs);
  2) Blackboard;
  3) Control component.

Knowledge sources

- Knowledge sources are independent modules that contain the knowledge needed for Problem solving.
- They don't need to know about the existence of the others, but they have to Understand the state of problem-solving process and the representation of relevant information on the blackboard.
- Knowledge sources can be represented with different kind of knowledge; they can include rule-based systems, case-based systems, neural networks, fuzzy logic systems, genetic algorithms, legacy software systems.

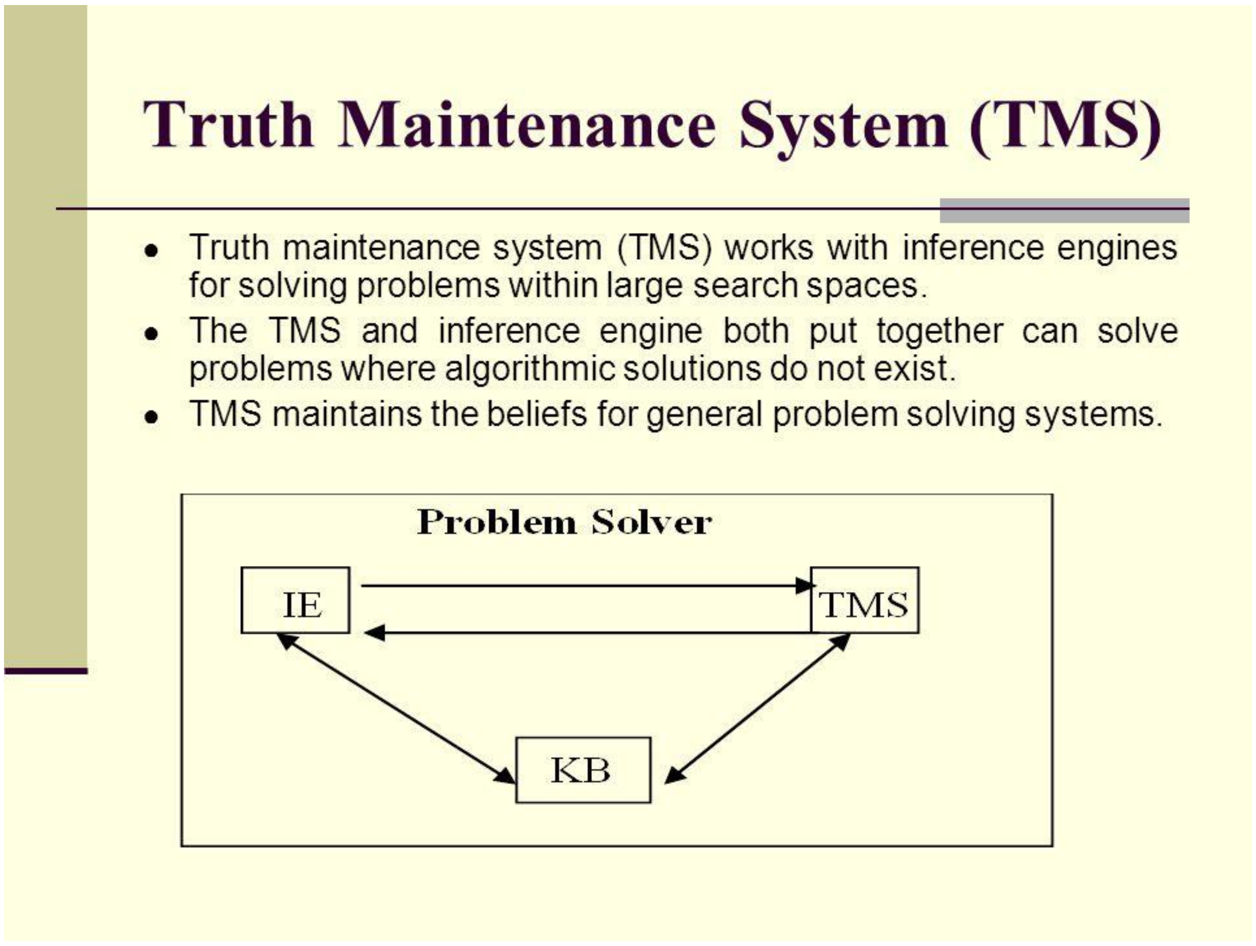


Blackboard

- Blackboard is used as a global database for sharing different information as input data,partial solutions, alternatives and final solutions.
- Blackboard applications tend to have complex blackboard structures, with multiple levels of analysis or abstraction.

Control

Control component makes runtime decisions which of knowledge sources to execute next for optimal problem solution.

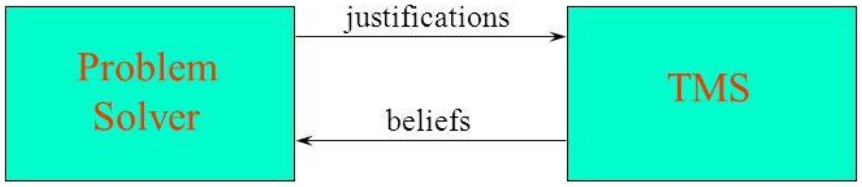


Truth Maintenance System (TMS)

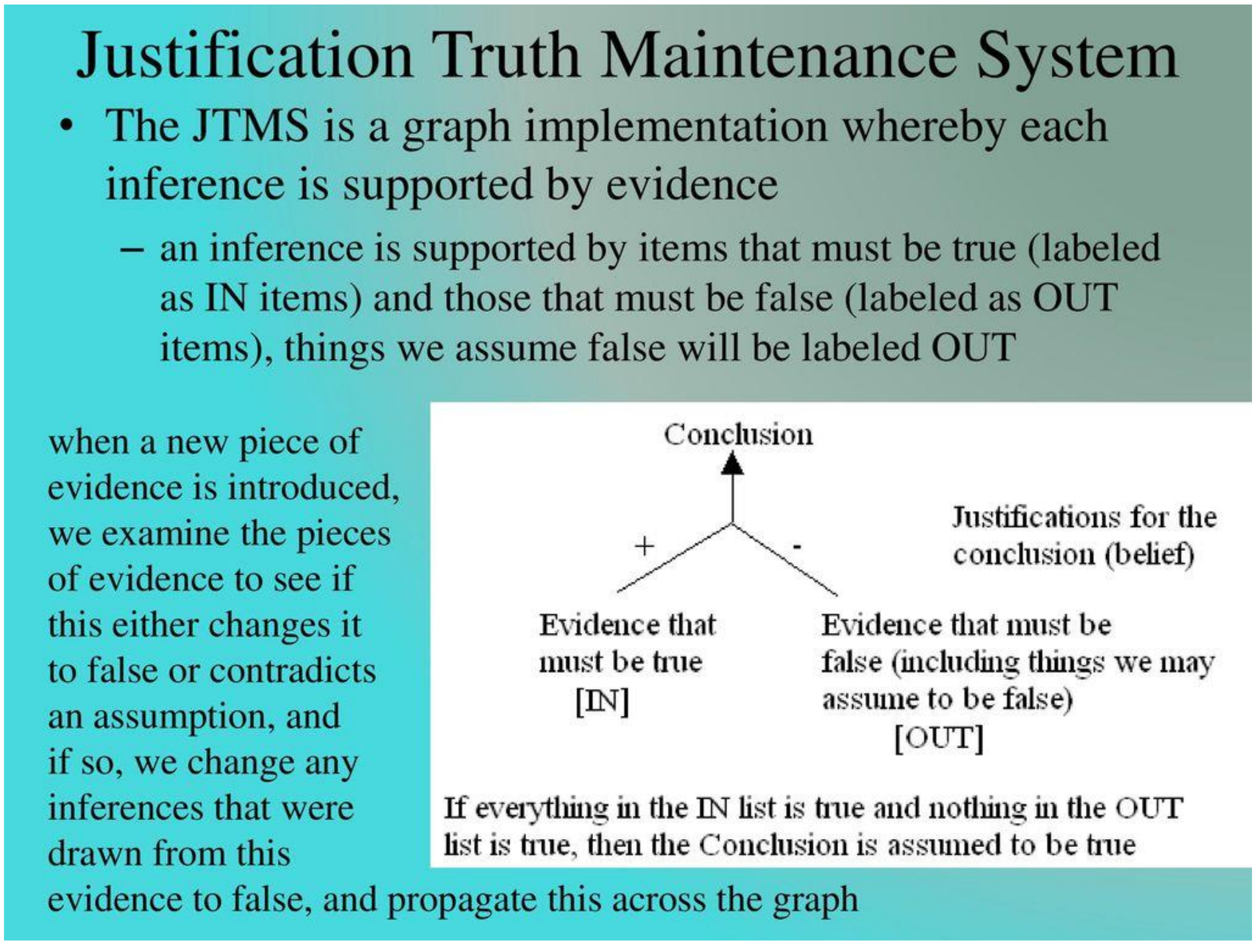
- Truth maintenance system (TMS) works with inference engines for solving problems within large search spaces.
- The TMS and inference engine both put together can solve problems where algorithmic solutions do not exist.
- TMS maintains the beliefs for general problem solving systems.

Problem Solver
IE
TMS
KB

Architecture of TMS-Based Agent

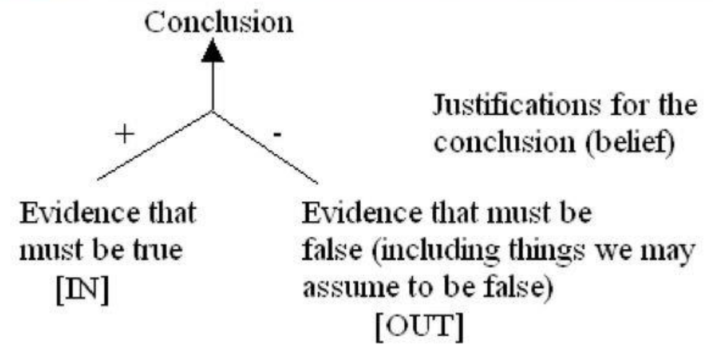


- The problem solver represents domain knowledge in the form of rules, procedures, etc. and chooses what to focus on next
- The TMS keeps track of the current state of the search for a solution. It uses constraint satisfaction to maintain consistency in the inferences made by the problem solver



Justification Truth Maintenance System

- The JTMS is a graph implementation whereby each inference is supported by evidence
  - an inference is supported by items that must be true (labeled as IN items) and those that must be false (labeled as OUT items), things we assume false will be labeled OUT


Conclusion
+
-
Justifications for the conclusion (belief)
Evidence that must be true [IN]
Evidence that must be false (including things we may assume to be false) [OUT]

If everything in the IN list is true and nothing in the OUT list is true, then the Conclusion is assumed to be true

when a new piece of evidence is introduced, we examine the pieces of evidence to see if this either changes it to false or contradicts an assumption, and if so, we change any inferences that were drawn from this evidence to false, and propagate this across the graph