Course Code : MCSE-003
Course Title : Artificial Intelligence and knowledge Management
Assignment Number : MCA(V)-E003/Assignment/2018-19
Maximum Marks : 100
Weightage : 25%
Last Date of Submission : 15th October, 2018 (For July Session)
15th April, 2019 (For January Session

Question 1: State and justify the validity of following inference rules
(i) Chain rule

(ii) Simplification

Ans. **Chain rule :-**

Chain rule

Generalizing the product rule leads to the **chain rule**. Let $E_1, E_2, \ldots . E_n$ be n events. The joint probability of all the n events is given by,

$$P(\bigcap_{i=1,..,n} E_i) = P(E_n | \bigcap_{i=1,..,n-1} E_i) * P(\bigcap_{i=1,..,n-1} E_i)$$

The chain rule can be used iteratively to calculate the joint probability of any no.of events.

Bayes' theorem

From the product rule, $P(X \cap Y) = P(X|Y)P(Y)$ and $P(Y \cap X) = P(Y|X)P(X)$. As $P(X \cap Y)$ and $P(Y \cap X)$ are both same,

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

where, $P(X) = P(X \cap Y) + P(X \cap Y^c)$ from sum rule.

(ii) Simplification:- In propositional logic, conjunction elimination (also called and elimination, ∧ elimination, or simplification) is a valid immediate, argument form and rule of inference which makes the inference that, if the conjunction A and B is true, then A is true, and B is true. The rule makes it possible to shorten longer proofs by deriving one of the conjuncts of a conjunction on a line by itself.

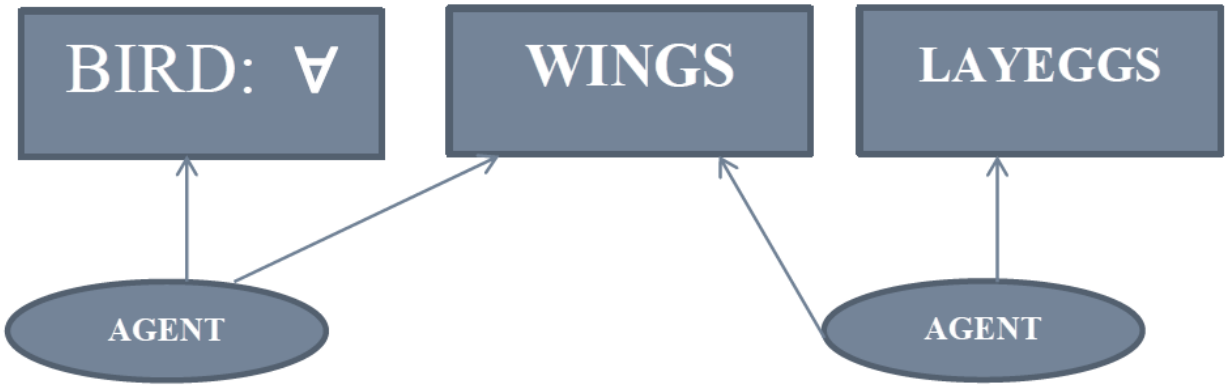
An example in English:
It's raining and it's pouring.
Therefore it's raining.
The rule consists of two separate sub-rules, which can be expressed in formal language as:

{\displaystyle {\frac {P\land Q}{\therefore P}}}
and
{\displaystyle {\frac {P\land Q}{\therefore Q}}}

The two sub-rules together mean that, whenever an instance of "{\displaystyle P\land Q}" appears on a line of a proof, either "{\displaystyle P}" or "{\displaystyle Q}" can be placed on a subsequent line by itself. The above example in English is an application of the first sub-rule.

Question 2: Transform the FOPL statement given below into equivalent conceptual graph.
∀ x (Has wings (x) Λ Layseggs (x) is_Bird (x))

Ans.



Question 3: Determine whether each of the following sentences are satisfactory, contradictory or valid
(i) P Λ Q) V ~ (P Λ Q)
(ii) (P Q) ~ P
Ans.

Ans 3

(i) $(P \wedge Q) \vee \sim (P \wedge Q)$

P	Q	$P \wedge Q$	$\sim P \wedge Q$	$(P \wedge Q) \vee \sim (P \wedge Q)$
0	0	False (0)	True	True
0	1	False (0)	True	True
1	0	False (0)	True	True
1	1	True (1)	False	True

A formula is said to be valid if and only if it is true under all its interpretation. Then $(P \wedge Q) \vee \sim$ $(P \wedge Q)$ is valid statement.

(ii) $(P \rightarrow Q) \rightarrow \sim P$

P	Q	$\sim P$	$P \rightarrow Q$	$(P \rightarrow Q) \rightarrow \sim P$
0	0	1	True	True
0	1	1	True	True
1	0	0	False	True
1	1	0	True	False

A formula is said to be inconsistent (or a contradiction) if & only if it is false under all interpretations

Question 4: Transform the following in to CNF (Any two)

(i) ~ (CD) V (C Λ D)

(ii) ~ (XY) Z

(iii)P (~ CQ R))

Ans 4 using method 1 : Truth table

(i) $\sim(C \to D) \vee (C \wedge D)$

C	D	$C \to D$	$\sim(C \to D)$	$C \wedge D$	$\sim(C \to D) \vee (C \wedge D)$
0	0	1	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	1	1	0	1	1

(ii) $\sim(x \to y) \to z$

x	y	z	$x \to y$	$\sim x \to y$	$\sim x - y \to z$
0	0	0	1	0	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	1	8	0
1	1	1	1		

Question 5: With the help of a suitable example, describe the "member" function of PROLOG. How Searching of a data in a list, recursively.

Ans.

Prolog uses brackets [...] as a list builder. The notation [X|Y] refers to a list whose first element is X and whose tail is Y. A finite list can be explicitly enumerated, such as [1,2,3,4]. The following three definitions should make sense to a Lisp programmer, where 'car' refers to the first element of a list, 'cdr' refers to the tail or rest of the list, and 'cons' is the list constructor.

car([X|Y],X).
cdr([X|Y],Y).
cons(X,R,[X|R]).

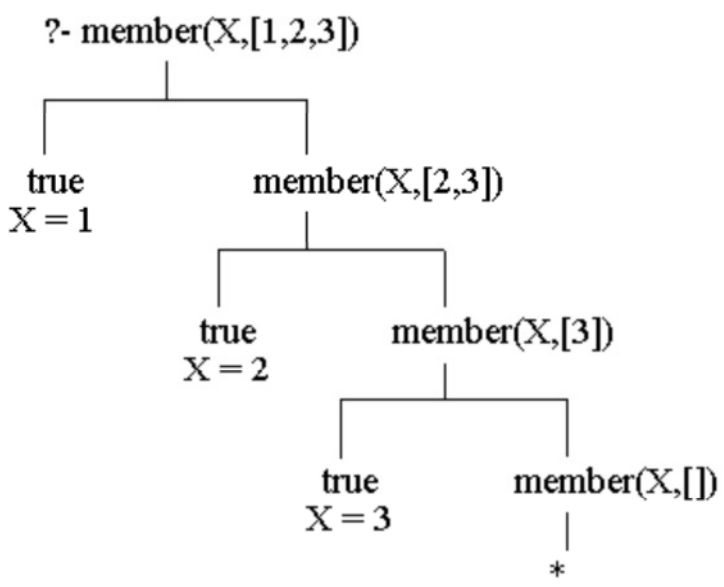
meaning ...

- *The head (car) of [X|Y] is X.*
- *The tail (cdr) of [X|Y] is Y.*
- *Putting X at the head and Y as the tail constructs (cons) the list [X|R].*

However, we will see that these explicit definitions are unneeded. A list whose head is X and whose tail is Y can just be referred to using the Prolog term [X|Y]. Conversely, if the list can be unified with the Prolog term '[X|Y]' then the first element of the list is bound to (unified with) X and the tail of the list is bound to Y.
Many of the predicates discussed in this section are "built-in" for many Prolog interpreters.
Consider the following definition of the predicate 'member/2'.
member(X,[X|R]).
member(X,[Y|R]) :- member(X,R).
One can read the clauses the following way, respectively:

- *X is a member of a list whose first element is X.*
- *X is a member of a list whose tail is R if X is a member of R.*

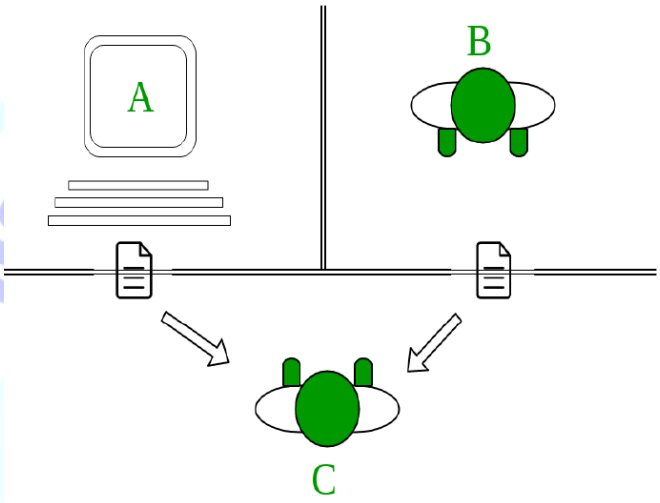
This program can be used in numerous ways. One can test membership:
?- member(2,[1,2,3]).
Yes
One can generate members of a list:
?- member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3 ;
No
Here is a derivation tree showing how this last goal generated all of the answers.



Question 6: What is Turing Test? If the machine passes Turing Test, does it mean that the system is intelligent? What are the associated problems with Turing Text? What are required improvements/advances to overcome these problems?

Ans.

The Turing test developed by Alan Turing (Computer scientist) in 1950. He proposed that "Turing test is used to determine whether or not computer (machine) can think intelligently like human"? Imagine a game of three players having two humans and one computer, an interrogator(as human) is isolated from other two players. The interrogator job is to try and figure out which one is human and which one is computer by asking questions from both of them. To make the things harder computer is trying to make the interrogator guess wrongly. In other words computer would try to indistinguishable from human as much as possible.



The "standard interpretation" of the Turing Test, in which player C, the interrogator, is given the task of trying to determine which player – A or B – is a computer and which is a human. The interrogator is limited to using the responses to written questions to make the determination

The conversation between interrogator and computer would be like this: **C(Interrogator):** Are you a computer?
A(Computer): No
C: Multiply one large number to another, 158745887 * 56755647
A: After a long pause, an incorrect answer!
C: Add 5478012, 4563145
A: (Pause about 20 second and then give as answer)10041157
If interrogator wouldn't be able to distinguish the answers provided by both human and computer then the computer passes the test and machine(computer) is considered as intelligent as human. In other words, a computer would be considered intelligent if it's conversation couldn't be easily distinguished from a human's. The whole conversation would be limited to a text-only channel such as a computer keyboard and screen.
He also proposed that by the year 2000 a computer "would be able to play the imitation game so well that an average interrogator will not have more than a 70-percent chance of making the right identification (machine or human) after five minutes of questioning." No computer has come close to this standard. But in year 1980, Mr. John searle proposed the "Chinese room argument". He argued that Turing test could not be used to determine "whether or not a machine is considered as intelligent like humans". He argued that any machine like ELIZA and PARRY could easily pass Turing

Test simply by manipulating symbols of which they had no understanding. Without understanding, they could not be described as "thinking" in the same sense people do. We will discuss more about this in next article.

Question 7: Transform the following conceptual graph in to FOPL statement

[PERSON: Anita]← (AGENT) ← [DEINK] →(OBJECT) →[Food: MILK] → ← (Instrument Glass)

Ans. (∃X) (∃Y) (GO(X)) ∧ Persion (Anita) ∧ (Ajent) (X,Deink) ∧ Food (X,Milk) ∧ Ins,Glass(Y,Glass)

Question 8: Describe „Means-ends Analysis" as problem solving technique.

Ans.

Most of the search strategies either reason forward of backward however, often a mixture o the two directions is appropriate. Such mixed strategy would make it possible to solve the major parts of problem first and solve the smaller problems the arise when combining them together. Such a technique is called "Means - Ends Analysis".
The means -ends analysis process centers around finding the difference between current state and goal state. The problem space of means - ends analysis has an initial state and one or more goal state, a set of operate with a set of preconditions their application and difference functions that computes the difference between two state a(i) and s(j). A problem is solved using means - ends analysis by

1. Computing the current state s1 to a goal state s2 and computing their difference D12.
2. Satisfy the preconditions for some recommended operator op is selected, then to reduce the difference D12.
3. The operator OP is applied if possible. If not the current state is solved a goal is created and means- ends analysis is applied recursively to reduce the sub goal.
4. If the sub goal is solved state is restored and work resumed on the original problem.

(the first AI program to use means - ends analysis was the GPS General problem solver)
means- ends analysis I useful for many human planning activities. Consider the example of planing for an office worker. Suppose we have a different table of three rules:

1. If in out current state we are hungry , and in our goal state we are not hungry , then either the "visit hotel" or "visit Canteen " operator is recommended.
2. If our current state we do not have money , and if in your goal state we have money, then the "Visit our bank" operator or the "Visit secretary" operator is recommended.
3. If our current state we do not know where something is , need in our goal state we do know, then either the "visit office enquiry" , "visit secretary" or "visit co worker " operator is recommended.

Question 9: Write a recursive program is LISP to find factorial of a number given by the user?

Ans.

Factorial

The factorial of a non-negative integer n , denoted by n! , is the product of all positive integers less than or equal to n.

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n-1)! \times n & \text{if } n > 0 \end{cases}$$

Example-

*4! = 4 * 3 * 2 * 1 = 24*

*similarly 6! = 6 * 5 * 4 * 3 * 2 * 1 = 720*

The value of 0! is 1, according to the convention for an empty product.

SOURCE CODE (IF INCLUDED)

```
(defun factorial (n)
(if (= n 1)
1
(* n (factorial (- n 1)))))
(factorial 4)
24
(factorial 6)
720
```

Question10: How a language for artificial intelligence differs from normal programming languages? Give name of three languages frequently used as programming language for developing Expert System .

Ans.

A typical program has three major segments: input, processing and output. So regular programming and Artificial Intelligence programming can be compared in terms of these three segments.

INPUT

In regular programming, input is a sequence of alphanumeric symbols presented and stored as per some given set of previously stipulated rules and that uses a limited set of communication media such as keyboard, mouse, disc, etc.

In Artificial Intelligence programming the input may be a sight, sound, touch, smell or taste. Sight means one dimensional symbols such as typed text, two dimensional objects or three dimensional scenes. Sound input include spoken language, music, noise made by objects. Touch include temperature, smoothness, resistance to pressure. Smell input include odors emanating from animate and inanimate objects. And taste input include sweet, sour, salty, bitter foodstuffs and chemicals.

PROCESSING

In regular programming, processing means manipulation of the stored symbols by a set of previously defined algorithms. In AI programming, processing includes knowledge representation and pattern matching, search, logic, problem solving and learning.

OUTPUT

In regular programming, output is a sequence of alphanumeric symbols, may be in a given set of colors, that represents the result of the processing and that is placed on such a medium as a CRT screen, paper, or magnetic disk.
In AI programming, output can be in the form of printed language and synthesized speech, manipulation of physical objects or locomotion i.e., movement in space.

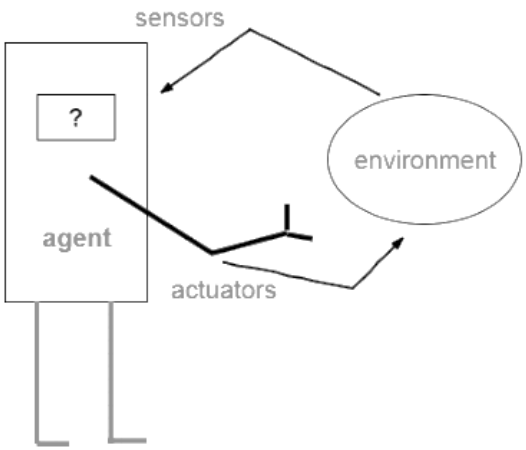
Three Languages which is used in Expert System:-
LISP (LIST Processing):
John McCarthy developed LISP in 1950, which stands of LISt processor. It supports symbolic manipulation and interactive trial and error style of programming. The most prolific advantage of LISP is that it has a set of primitive operator for carrying out deduction with sentences containing words representing predicates and their arguments and thus helps in implementing logical inferences.(Akerkar, 2007)
Prolog:
A PROLOG program consists of set of clauses. A clause is either a fact of a rule, which is used to indicate a relationship between elements. PROLOG tries to match the arguments of the query with the facts in the data base. This process is known as unification. If the unification succeeds, the variable is said to be instantiated.
The inference process for PROLOG programming is as follows. (Rolston, 1988)
Given a goal, PROLOG searches the database, starting at the top, for a fact that matches the goal. When PROLOG finds a match and instantiates the appropriate variables, it leaves a pointer where the match occurred. When a goal matches the head of a rule rather than a fact, the atoms within the body of the rule are treated as sub goals that must all be satisfied to prove that the head is satisfied.
Expert system tools:
The main task of expert system development tools is making development of an expert system much easier compared to programming language. Developers say that selecting the correct tool is a vital in design and development of an expert system. The reasons are manifold. They are: (Rolston, 1988)
They provide rich software development environment like structure editors, powerful debugging and tracing facility, multi windows, graphics etc.
Allows rapid prototyping because of incremental compilers, and automatic version control.
Defining model, knowledge representation and inference design are built into the tools.
Helps in maintaining the system and historical database.

Question11: What do you mean by term "Agents" in Artificial Intelligence? Classify the various types of agents.

Ans.

Artificial intelligence is defined as study of rational agents. A rational agent could be anything which makes decisions, like a person, firm, machine, or software. It carries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance). An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents. An agent is anything that can be viewed as :

- perceiving its environment through sensors and
- acting upon that environment through actuators



To understand the structure of Intelligent Agents, we should be familiar with Architecture and Agent Program. Architecture is the machinery that the agent executes on. It is a device with sensors and actuators, for example : a robotic car, a camera, a PC. Agent program is an implementation of an agent function. An agent function is a map from the percept sequence(history of all that an agent has perceived till date) to an action.

Examples of Agent:-
A software agent has Keystrokes, file contents, received network packages which act as sensors and displays on the screen, files, sent network packets acting as actuators.
A Human agent has eyes, ears, and other organs which act as sensors and hands, legs, mouth, and other body parts acting as actuators.
A Robotic agent has Cameras and infrared range finders which act as sensors and various motors acting as actuators.

Types of Agents

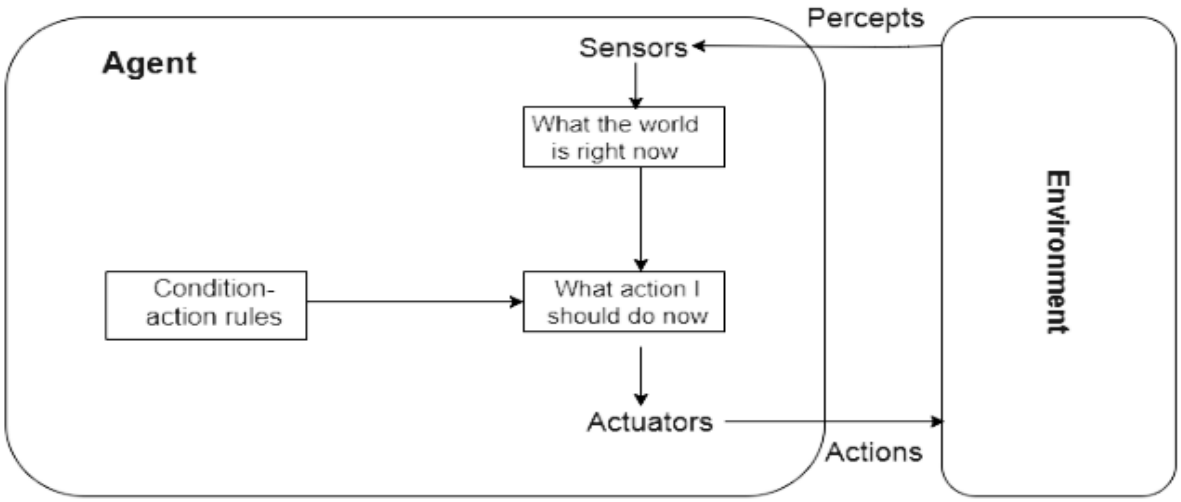
Agents can be grouped into four classes based on their degree of perceived intelligence and capability :

Simple reflex agents :-

Simple reflex agents ignore the rest of the percept history and act only on the basis of the current percept. Percept history is the history of all that an agent has perceived till date. The agent function is based on the condition-action rule. A condition-action rule is a rule that maps a state i.e, condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable. It may be possible to escape from infinite loops if the agent can randomize its actions. Problems with Simple reflex agents are :

- Very limited intelligence.
- No knowledge of non-perceptual parts of state.

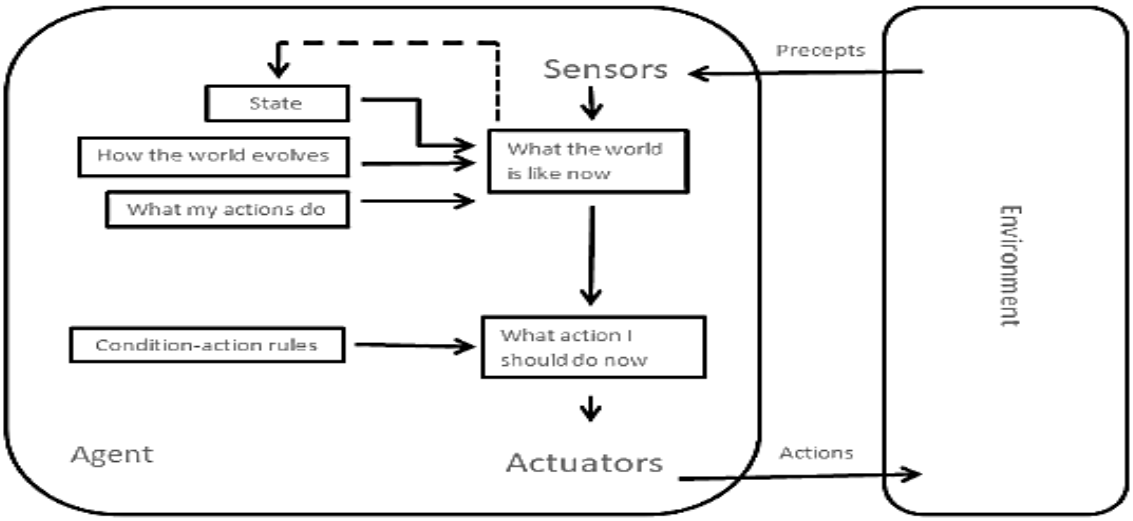
- Usually too big to generate and store.
- If there occurs any change in the environment, then the collection of rules need to be updated.



Model-based reflex agents

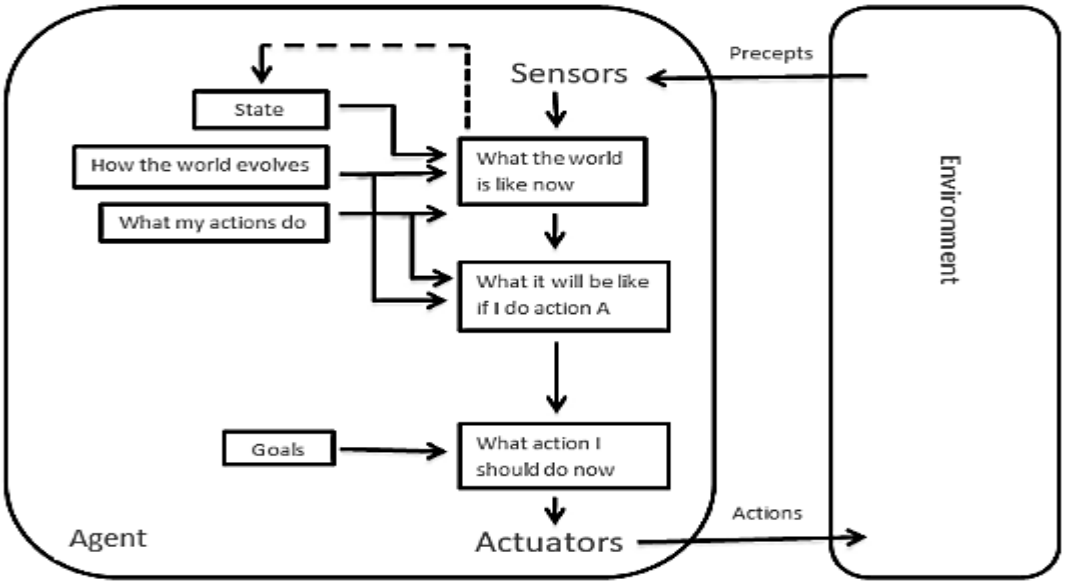
It works by finding a rule whose condition matches the current situation. A model-based agent can handle partially observable environments by use of model about the world. The agent has to keep track of internal state which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen. Updating the state requires the information about :

- how the world evolves in-dependently from the agent, and
- how the agent actions affects the world.



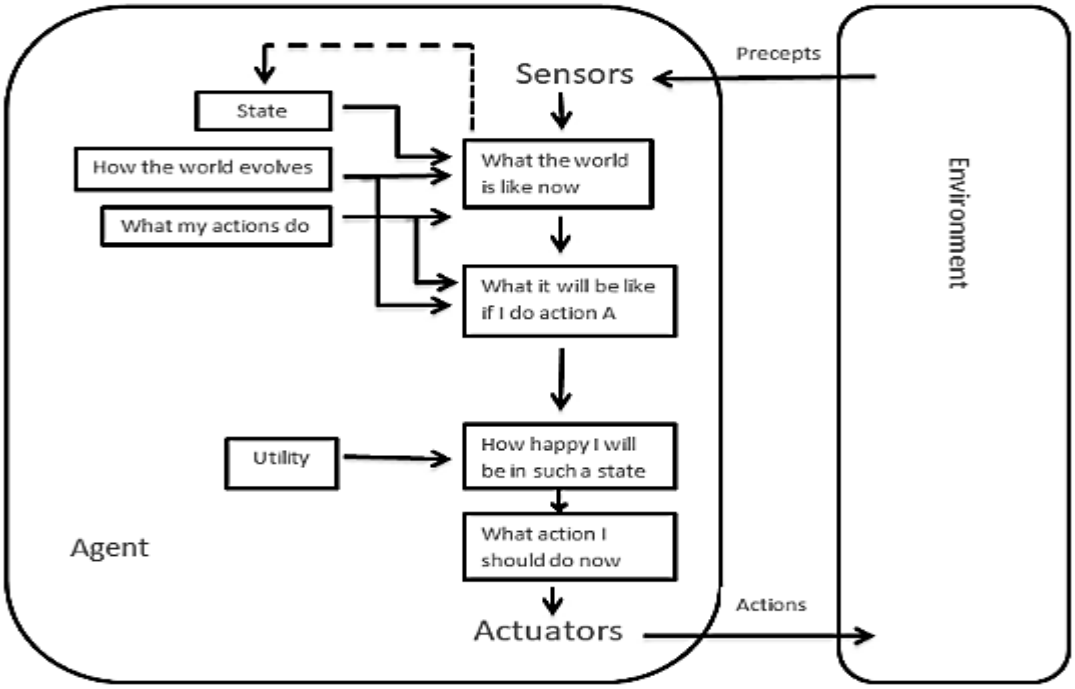
Goal-based agents

These kind of agents take decision based on how far they are currently from their goal(description of desirable situations). Their every action is intended to reduce its distance from goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require search and planning. The goal based agent's behavior can easily be changed.



Utility-based agents

The agents which are developed having their end uses as building blocks are called utility based agents. When there are multiple possible alternatives, then to decide which one is best, utility based agents are used.They choose actions based on a preference (utility) for each state. Sometimes achieving the desired goal is not enough. We may look for quicker, safer, cheaper trip to reach a destination. Agent happiness should be taken into consideration. Utility describes how "happy" the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.



Question12: Briefly describe the term "Truth Maintainance System – TMS".

Ans. **Truth Maintenance System (TMS):**
The logics are known as monotonic logics. The conclusions derived using such logics are valid deductions, and they remain so for all times. Adding new axioms increases the amount of knowledge contained in the knowledge base. Therefore, the set of facts and inferences in such systems can only grow larger; they cannot be reduced; that is, they increase monotonically.

The form of reasoning referred to above, on the other hand, is non-monotonic. New facts become known which can contradict and can invalidate the old knowledge. The old knowledge is retracted causing other dependent knowledge to become invalid, thereby requiring further retractions. The retractions lead to a shrinkage or growth of knowledge base, called non-monotonic growth in the knowledge, at times.

This can be illustrated by a real-life situation. Suppose a young boy Sahu enjoys seeing movie in a cinema hall on the first day of its release. He insists upon his grand father, Mr. Girish in accompanying him. Mr. Girish has agreed to accompany Sahu there on the following Friday evening. On the Thursday, when forecasts predicted heavy snow.

Now, believing the weather would discourage most senior citizens, Girish changed his mind of joining Mr. Sahu. But, unexpectedly, on the given Friday, the forecasts proved to be false; so Mr. Girish once again went to see movie. This is the case of non-monotonic reasoning.

It is not reasonable to expect that all the knowledge needed for a set of tasks could be acquired, validated, and loaded into the system at the outset. More typically, the initial knowledge will be incomplete, contain redundancies, inconsistencies, and other sources of uncertainty. Even if it were

possible to assemble complete, valid knowledge initially, it probably would not remain valid forever, more so in a continually changing environment.

In an attempt to model real-world, commonsense reasoning, researchers have proposed extensions and alternatives to traditional logics such as Predicate Logic and First Order Predicate Logic. The extensions accommodate such real time forms of uncertainty and non-monotony as experienced by our subject, Mr. Girish.

We now give a description of Truth maintenance systems (TMS), which have been implemented to permit a form of non-monotonic reasoning by permitting the addition of changing (even contradictory) statements to a knowledge base. Truth maintenance system (also known as belief revision system) is a companion component to inference system.

The main object of the TMS is the maintenance of the knowledge base used by the problem solving system and not to perform any inference. As such, it frees the problem solver from any concerns of knowledge consistency check when new knowledge gets added or deleted and allows it to concentrate on the problem solution aspects.

The TMS also gives the inference component the latitude to perform non-monotonic inferences. When new discoveries are made, this more recent information can displace the previous conclusions which are no longer valid.

In this way, the set of beliefs available to the problem solver will continue to be current and consistent.

Fig. 7.1 illustrates the role played by the TMS as a part of the problem solving system. The Inference Engine (IE) from the expert system or decision support system solves domain specific problems based on its current belief set, maintained by the TMS. The updating process is incremental. After each inference, information is exchanged between the two components the IE and the TMS.

The IE tells the TMS what deductions it has made. The TMS, in turn, asks questions about current beliefs and reasons for failure of earlier statements. It maintains a consistent set of beliefs for the IE to work with when the new knowledge is added or removed.

For example, suppose the knowledge base (KB) contained only the propositions P and P → Q, and modus ponens. From this, the IE would rightfully conclude Q and add this conclusion to the KB. Later, if it was learned that ~P become true it would be added to the KB resulting in P becoming false leading to a contradiction. Consequently, it would be necessary to remove P to eliminate the inconsistency. But, with P now removed, Q is no longer a justified belief. It too should be removed. This type of belief revision is the job of the TMS.

Actually, the TMS does not discard conclusions like Q as suggested. That could be wasteful, since P may again become valid, which would require that Q and facts justified by Q be re-derived. Instead, the TMS maintains dependency records for all such conclusions.

These records determine which set of beliefs are current and are to be used by the IE. Thus, Q would be removed from the current belief set by making appropriate updates to the records and not by erasing Q. Since Q would not be lost, its re-derivation would not be necessary if and when P became valid once again.

The TMS maintains complete records of reasons or justifications for beliefs. Each proposition or statement having at least one valid justification is made a part of the current belief set. Statements lacking acceptable justifications are excluded from this set.

When a contradiction is discovered, the statements responsible for the contradiction are identified and an appropriate one is retracted. This in turn may result in other reactions and additions. The procedure used to perform this process is called dependency directed back tracking which will be explained shortly.

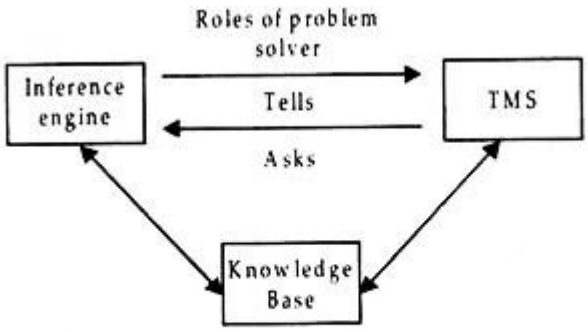


Fig. 7.1. *Architecture of the problem solver with a TMS.*

Question13: Explain the following logic concepts, if required use suitable examples (Any two):
(i) Modus Tollens
(ii) Satisfiable statement
(iii) Resolution principle in proposition logic

Ans. **(ii)** I tend to see these words a lot in Discrete Mathematics. I assumed these were just simple words until I bumped into a question.
Is the following proposition **Satisfiable?** Is it **Valid**? (P→Q)⇔(Q→R)(P→Q)⇔(Q→R)
Then I searched in the net but in vain. So I'm asking here. What do you mean by Satisfiable and Valid? Please explain.

(iii) Propositional Resolution works only on expressions in *clausal form*. Before the rule can be applied, the premises and conclusions must be converted to this form. Fortunately, as we shall see, there is a simple procedure for making this conversion.
A *literal* is either an atomic sentence or a negation of an atomic sentence. For example, if p is a logical constant, the following sentences are both literals.

$$p$$
$$\neg p$$

A *clausal sentence* is either a literal or a disjunction of literals. If p and q are logical constants, then the following are clausal sentences.

$$p$$
$$\neg p$$
$$\neg p \vee q$$

A *clause* is the set of literals in a clausal sentence. For example, the following sets are the clauses corresponding to the clausal sentences above.

$$\{p\}$$
$$\{\neg p\}$$

$\{\neg p, q\}$

Note that the empty set {} is also a clause. It is equivalent to an empty disjunction and, therefore, is unsatisfiable. As we shall see, it is a particularly important special case.

The conversion rules are summarized below and should be applied in order.

1. Implications (I):

$\varphi \Rightarrow \psi \quad \rightarrow \quad \neg\varphi \vee \psi$

$\varphi \Leftarrow \psi \quad \rightarrow \quad \varphi \vee \neg\psi$

$\varphi \Leftrightarrow \psi \quad \rightarrow \quad (\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi)$

2. Negations (N):

$\neg\neg\varphi \quad \rightarrow \quad \varphi$

$\neg(\varphi \wedge \psi) \quad \rightarrow \quad \neg\varphi \vee \neg\psi$

$\neg(\varphi \vee \psi) \quad \rightarrow \quad \neg\varphi \wedge \neg\psi$

3. Distribution (D):

$\varphi \vee (\psi \wedge \chi) \quad \rightarrow \quad (\varphi \vee \psi) \wedge (\varphi \vee \chi)$

$(\varphi \wedge \psi) \vee \chi \quad \rightarrow \quad (\varphi \vee \chi) \wedge (\psi \vee \chi)$

$\varphi \vee (\varphi 1 \vee \ldots \vee \varphi n) \quad \rightarrow \quad \varphi \vee \varphi 1 \vee \ldots \vee \varphi n$

$(\varphi 1 \vee \ldots \vee \varphi n) \vee \varphi \quad \rightarrow \quad \varphi 1 \vee \ldots \vee \varphi n \vee \varphi$

$\varphi \wedge (\varphi 1 \wedge \ldots \wedge \varphi n) \quad \rightarrow \quad \varphi \wedge \varphi 1 \wedge \ldots \wedge \varphi n$

$(\varphi 1 \wedge \ldots \wedge \varphi n) \wedge \varphi \quad \rightarrow \quad \varphi 1 \wedge \ldots \wedge \varphi n \wedge \varphi$

4. Operators (O):

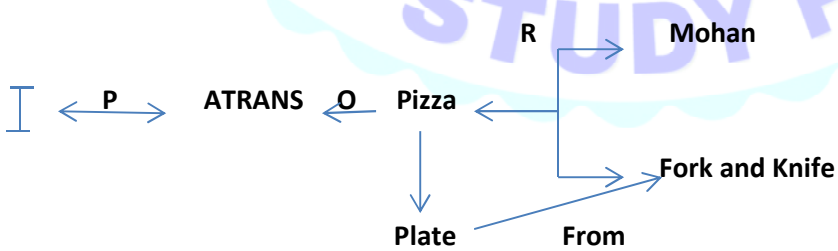
$\varphi 1 \vee \ldots \vee \varphi n \quad \rightarrow \quad \{\varphi_1, \ldots, \varphi_n\}$

$\varphi 1 \wedge \ldots \wedge \varphi n \quad \rightarrow \quad \{\varphi_1\}, \ldots, \{\varphi_n\}$

Question14: Give conceptual dependency representation of the sentence give below: "Mohan will eat pizza from the plate with fork and knife"

Ans.

"Mohan will eat pizza from the plate with fork and knife."



Question15: Compare and contrast the following:
(i) Frames and scripts
(ii) Informed search and uniformed search
(iii) Abductive inference and Analogical inference
(iv) A* algorithm and AO* algorithm

Ans. **(i)**

Frames are very similar to the class diagrams we draw to represent an object oriented algorithm and frames necessarily form a collection of slots (or objects). A single frame taken is rarely useful. The below diagram gives an overview of how frames can be used to represent the hotel and objects found in a hotel. Each square box forms a slot and it includes the fillers in them and there is a relationship between objects.

A script is a structure that describes a stereotyped sequence of events in a particular context. There are a few important components of scripts -

1. Entry conditions - What are the states of different objects at the beginning.
2. Roles - What are the different slots(people) involved in the events.
3. Props - What are the different slots(objects) involved in the events.
4. Tracks - What is the sequences of events that is supposed to happen.
5. Scenes - The formal representation of events.
6. Results - Conditions that will, in general be true after the events described in the scripts have occurred.

(ii) ***Heuristic/Informed Seach***: searching with information. **example**: A* Algorithm. We choose our next state based on cost and 'heuristic information' with heuristic function.
Case Example: find the shortest path. 1- **Blind search**, we just trying all location (brute force). 2- **Heuristic**, say we have information about the distance between start point and each available location. We will use that to determine next location.

Blind/Uniformed Search: searching without information. **example**: BFS (one of blind search method). We just generate all successor state (child node) for the current state (current node) and find is there a goal state among them, if not exist we will generate one of child node's successor and so on. Because we don't have information, so just generate all.

(iii) Abductive reasoning (also called abduction, abductive inference,or retroduction) is a form of logical inference which starts with an observation or set of observations then seeks to find the simplest and most likely explanation. This process, unlike deductive reasoning, yields a plausible conclusion but does not positively verify it. Abductive conclusions are thus qualified as having a remnant of uncertainty or doubt, which is expressed in retreat terms such as "best available" or "most likely". One can understand abductive reasoning as inference to the best explanation,[3] although not all uses of the terms abduction and inference to the best explanation are exactly equivalent.

Analogy (from Greek ἀναλογία, analogia, "proportion", from ana- "upon, according to" [also "against", "anew"] + logos "ratio" [also "word, speech, reckoning"]) is a cognitive process of transferring information or meaning from a particular subject (the analog, or source) to another (the target), or a linguistic expression corresponding to such a process. In a narrower sense, analogy is an inference or an argument from one particular to another particular, as opposed to deduction,

induction, and abduction, in which at least one of the premises, or the conclusion, is general rather than particular in nature.

(iv) 1. a* is a computer algorithm which is used in path finding and graph traversal. It is used in the process of plotting an efficiently directed path between a number of points called nodes.
2. In a* algorithm you traverse the tree in depth and keep moving and adding up the total cost of reaching the cost from the current state to the goal state and add it to the cost of reaching the current state.

1. In ao* algorithm you follow a similar procedure but there are constraints traversing specific paths.
2. When you traverse those paths, cost of all the paths which originate from the preceding node are added till that level, where you find the goal state regardless of the fact whether they take you to the goal state or not.

Question16: Define following properties of propositional statement:
(i) Satisfiable
(ii) Contradiction
(iii) Valid
(iv) Equivalent
(v) Logical consequtence
Ans.

(i) Satisfiable
In mathematical logic, satisfiability and validity are elementary concepts of semantics. A formula is satisfiable if it is possible to find an interpretation (model) that makes the formula true.[1] A formula is valid if all interpretations make the formula true. The opposites of these concepts are unsatisfiability and invalidity, that is, a formula is unsatisfiable if none of the interpretations make the formula true, and invalid if some such interpretation makes the formula false. These four concepts are related to each other in a manner exactly analogous to Aristotle's square of opposition.

The four concepts can be raised to apply to whole theories: a theory is satisfiable (valid) if one (all) of the interpretations make(s) each of the axioms of the theory true, and a theory is unsatisfiable (invalid) if all (one) of the interpretations make(s) each of the axioms of the theory false.

(ii) Contradiction

Tautology/Contradiction/Contingency.

DEFINITION 2.1.1. *A* **tautology** *is a proposition that is always true.*

EXAMPLE 2.1.1. $p \vee \neg p$

DEFINITION 2.1.2. *A* **contradiction** *is a proposition that is always false.*

EXAMPLE 2.1.2. $p \wedge \neg p$

DEFINITION 2.1.3. *A* **contingency** *is a proposition that is neither a tautology nor a contradiction.*

EXAMPLE 2.1.3. $p \vee q \rightarrow \neg r$

(iii) Valid

- Many claims come to us as the conclusion of arguments. By "conclusion," we mean the claim that the argument is meant to defend.
- We will understand an argument as a list of claims, one of which is the conclusion, and the other claims of which are offered as reasons to believe the conclusion is true. We will call these other claims "premises."
- We need to determine what makes an argument good. Look at some examples.

1. Premise 1: If Nixon was President, then Nixon was Commander in Chief.
2. Premise 2: Nixon was President.
3. Conclusion: Nixon was Commander in Chief.

1. Premise 1: If Lincoln was an organism from deep in the sea, then Lincoln had three eyes.
2. Premise 2: Lincoln was an organism from deep in the sea.
3. Conclusion: Lincoln had three eyes.

1. Premise 1: If Lincoln was President, then Lincoln had at least one portrait made of him.
2. Premise 2: Lincoln had at least one portrait made of him.
3. Conclusion: Lincoln was President.

(iv) Equivalent

In logic, statements {\displaystyle } and {\displaystyle q} are logically equivalent if they have the same logical content. That is, if they have the same truth value in every model (Mendelson 1979:56). The logical equivalence of {\displaystyle p} and {\displaystyle q} is sometimes expressed as {\displaystyle p\equiv q}, {\displaystyle {\textsf {E}}pq}, or {\displaystyle p\iff q}. However, these symbols are also used for material equivalence. Proper interpretation depends on the context. Logical equivalence is different from material equivalence, although the two concepts are closely related.

Some basic established logical equivalences are tabulated below-

$$p \rightarrow q \equiv \neg p \vee q$$
$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$
$$p \wedge q \equiv \neg(q \rightarrow \neg p)$$
$$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$$
$$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$$
$$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$$
$$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$$

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$
$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$
$$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$
$$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$$

(v) Logical consequtence

Logical consequences involve action taken by the parent. No consequence should ever place a child at risk for injury.

Examples of Logical Consequences

- "in control" temper tantrum (yelling and pounding on floor; no one is getting hurt)-- leave area of tantrum, i.e., tantrum does not even get attention, which was the goal
- leaves toys all over—toys get a timeout
- misbehaves at dinner-- leaves the table
- do not give choices when there are none: "It is time for bed" not "it is time for bed, OK?"
- no dinner, no dessert (or late night snacks for that matter). It should be child's choice to eat or not eat. Child should eat reasonable dinner to get "treats."
- breaking curfew results in grounding
- annoying or gets into things while you are on the phone-- grounding to room/chair time (anticipate, by planning activity for child while you are on the phone and avoid need for any consequences, everybody will be happier!)
- do not threaten consequences you cannot or will not enforce, e.g., "I will throw out your toys if you do not pick them up." Financially this is not a good idea and you are not likely to follow through with the threat. Even if you are, the message to the child is not very practical and is actually reckless.
- ride bike outside area permitted--bike gets grounded
- does not wear bike helmet--bike gets grounded
- stays out beyond curfew hours--curfew time becomes earlier
- stealing--1] pays victim out of own pocket; 2] returns item with apology
- lying--lowers trust, adult must "confirm" information child provides until child re-earns trust
- speaks with disrespect--do not respond, ignore the child's presence; "I will not speak with you unless you are respectful" (being angry is not disrespectful; name calling is disrespectful)
- destroys property--pays for it out of allowance, work, etc.
- sneaks out of house--grounding for reasonable period of time
- if clothes do not get to the hamper—they do not get washed. May have to wear soiled, wrinkled clothes toschool as a result.

*In the eyes of the child, parents are "mean" when the child discovers that their parents MEAN what they say!

"I will treat you as responsible as you behave." If kids behave properly than they earn the privilege of greater independence and freedom, i.e., less adult supervision. On the other hand if they act irresponsibly, than they should expect to be treated accordingly. For example, their bike gets left outside and is stolen (parents refusing to replace bike, child having to save money for replacement is a logical consequence as child is not demonstrating responsibility.) Consequences are what influence most of what we do on a daily basis. Unpleasant outcomes usually keep us from repeating the same decision. (Get a speeding ticket, we slow down. Spill grape juice on living room carpet, we don't drink grape juice in the living room. Yell at boss, get fired, you don't yell at the next boss.) Consequences are what help us become responsible people. We do the right things because we don't like the outcomes if we don't. If we make bad choices and there are no bad outcomes we learn nothing and continue to make the bad choices. Say you make a bad choice; someone

Question17: What is meant by 'Closed Word Assumption'? Where is it used in AI?

Ans. The **closed-world assumption** (CWA), in a formal system of logic used for knowledge representation, is the presumption that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false. The same name also refers to a logical formalization of this assumption by Raymond Reiter.The opposite of the closed-world assumption is the open-world assumption (OWA), stating that lack of knowledge does not imply falsity. Decisions on CWA vs. OWA determine the understanding of the actual semantics of a conceptual expression with the same notations of concepts. A successful formalization of natural language semantics usually cannot avoid an explicit revelation of whether the implicit logical backgrounds are based on CWA or OWA.
Negation as failure is related to the closed-world assumption, as it amounts to believing false every predicate that cannot be proved to be true.

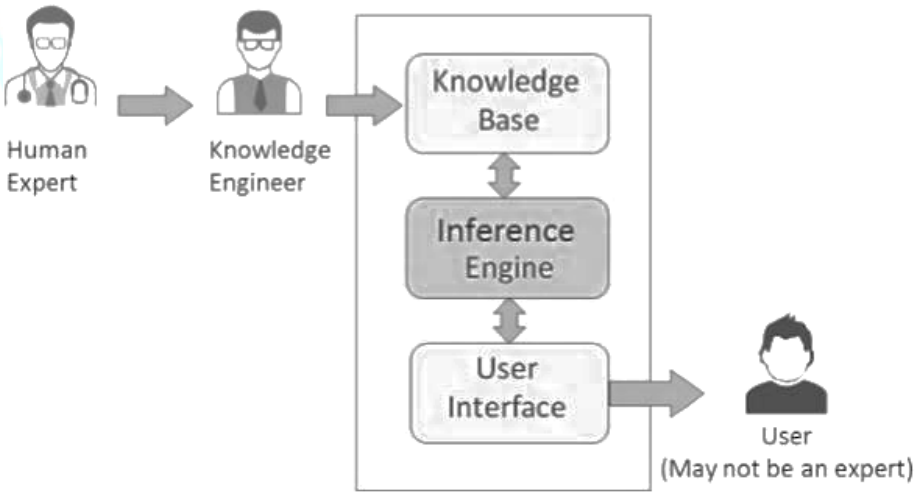
Question18: Write short notes on any two of the following:
(i) Expert systems (ii) Non Deductive Inference rule
(iii) Methods to deal with Uncertainty in knowledge systems

Ans. **(i) Expert systems**
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 of Expert Systems

- High performance
- Understandable
- Reliable
- Highly responsive
- Capabilities of Expert Systems
- The expert systems are capable of −
- 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
-
- They are incapable of –
- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge
- Components of Expert Systems
- The components of ES include –
- Knowledge Base
- Inference Engine
- User Interface

Let us see them one by one briefly –



(ii) **Non Deductive Inference rule**

Definition: A non-deductive argument is an argument for which the premises are offered to provide probable – but not conclusive – support for its conclusions.
In a good non-deductive argument, if the premises are all true, you would rightly expect the conclusion to be true also, though you would accept that it may be false.
If you like, think of non-deductive arguments in terms of bets. If the premises of a good non-deductive argument are true, then you would be happy to bet that the conclusion is also true. The argument would have provided you with the confidence that your bet is a sensible one, but – since it is a bet, after all – you would accept that the conclusion may turn out false and you may lose.

Question19: Explain the difference between Forward and Backward Chaining. Under which situation which mechanism is best to use, for a given set of problems?

Ans.

Forward chaining :-
1. It is also known as data driven inference technique.
2. Forward chaining matches the set of conditions and infer results from these conditions. Basically, forward chaining starts from a new data and aims for any conclusion.
3. It is bottom up reasoning.
4. It is a breadth first search.
5. It continues until no more rules can be applied or some cycle limit is met.
6. For example: If it is cold then I will wear a sweater. Here "it is cold is the data" and "I will wear a sweater"is a decision. It was already known that it is cold that is why it was decided to wear a sweater, This process is forward chaining.
7. It is mostly used in commercial applications i.e event driven systems are common example of forward chaining.
8. It can create an infinite number of possible conclusions.

Backward chaining
1. It is also called as goal driven inference technique.
2. It is a backward search from goal to the conditions used to get the goal. Basically it starts from possible conclusion or goal and aims for necessary data.
3. It is top down reasoning.
4. It is a depth first search.
5. It process operations in a backward direction from end to start, it will stop when the matching initial condition is met.
6. For example: If it is cold then I will wear a sweater. Here we have our possible conclusion "I will wear a sweater". If I am wearing a sweater then it can be stated that it is cold that is why I am wearing a sweater. Hence it was derived in a backward direction so it is the process of backward chaining.
7. It is used in interrogative commercial applications i.e finding items that fulfill possible goals.
8. Number of possible final answers is reasonable.

best to use In the old old expert systems days they used to say forward chaining was good for looking around (checking for what could be) while backward chaining was good for confirming (checking if "it" really is).

Question20: Express the following knowledge as a semantic network structure with Interconnected nodes and labeled arcs. "Ram is Vice President of ABC Company. He is married to Raj and has a male child RamRaj. RamRaj Goes to school. Ram plays golf and owns a silver color German made car Mercedez Benz"

Ans.