Artificial Intelligence (CSC261)

Note By:

- **Roshan Bist**
- SNSC, Mahendranagar
- https://www.facebook.com/roshanbist.roshanbist.3
- https://www.facebook.com/notejunction/
- notejunction360@gmail.com

UNIT-1
Introduction

⊛ **Intelligence:** It is the capacity to learn and solve problems. It is the to solve novel problems, act rationally and act like humans.

It is the part of computer science concerned with designing intelligence computer systems that is, systems exhibit the characteristics we associate with intelligence in human behaviour.

⊛ **Artificial Intelligence (AI):** In short we can say that artificial intelligence is giving machines ability to perform tasks normally associated with human intelligence. With AI issues like deduction, reasoning, problem solving, knowledge representation, planning, learning, natural language processing came into existence.

But there are four main different approaches for defining artificial intelligence in eight different ways by different people with different methods based on thinking humanly, acting humanly, thinking rationally and acting rationally as follows:-

a) Thinking Humanly →

1). "The exciting new effort to make computers think... machines with minds, in the full and literal sense." (Haugeland, 1985)

2). "The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." (Bellman, 1978).

b) Acting Humanly

3). "The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990).

4). "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991).

deals more

c) Thinking Rationally:

5). "The study of mental faculties through the use of computational methods." (Charniak and Mc Dermott, 1985).

6). "The study of computations that make it possible to preceive, reason and act." (Winston, 1992).

d) Acting Rationally:

7). "Computational Intelligence is the study of design of intelligent agents". (Poole et al, 1998).

8). "AI... is concerned with intelligent behaviour in artifacts." (Nilsson, 1998).

⇒ These 4 different approaches in more detail are as follows:- [Imp]

Thinking Humanly: (The Cognitive modeling approach)

If we are going to say that a given program thinks like a human, then we must have some way of determining how humans think. We need to go inside the actual workings of human minds. There are three ways to do this:

i) Through introspection (i.e, trying to catch our own thoughts as they go by).

ii) Through psychological experiments (i.e, observing a person in action).

iii). Through Brain imaging (i.e, observing the brain in action).

Once we have a sufficiently precise theory of mind, it becomes possible to express the theory as computer program. If the programs input-output behaviour matches corresponding human-behaviour that is evidence that some of the programs mechanism could also be operating in humans.

Acting Humanly: (The turning test approach) [V. Imp]

A computer could be called intelligent if it passes the test of a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer. The computer would need to possess the following capabilities:

i) natural language processing → to enable it to communicate successfully in English.

ii) knowledge representation → to store what it knows or hears.

iii) automated reasoning → to use the stored information to answer questions and to draw new conclusions.

iv) machine learning → to adopt new circumstances and to detect and extrapolate patterns.

Thinking rationally: (The "laws of thought" approach)

Aristotle was one of the first to attempt to codify "right thinking", that is irrefutable reasoning process. He gave syllogisms that always yeilded correct conclusion when correct premises are given.

For example:- If Roshan is man.
All men are mortal.
Then ⇒ Roshan is mortal.

These law of thought were supposed to govern the operation of mind: this study initiated the field of logic. The logicist tradition in AI hopes to create intelligent systems using logic programming. However there are two obstacles to this approach.

i) First it is not easy to take informal knowledge and state in the formal terms required by logical notation, particularly when knowledge is not 100% certain.

ii) Second is solving problem "in principle" is different from doing it in practice.

Acting rationally: (The rational agent approach)

An agent is just something that acts but a rational agent is one that acts so as to achieve the best outcome. Computer agent is expected to have autonomous control, preceiving their environment, persisting over a prolonged period of time, adopting to change and capable of taking on another's goal.

In the "laws of thought" approach to AI, the emphasis

was given to correct inferences. Making correct inferences is sometimes part of being a rational agent, because one way to act rationally is to reason logically to the conclusion and act on that conclusion.

Advantages:-

- → It is more general than laws of thought approach because correct inference is just one of several mechanisms for achieving rationality.
- → It is more amenable to scientific development than other approaches based on human behaviour or human thought.

⊛. History of AI:

Warren Mc Culloch and Walter Pitts (1943): a model of artificial boolean neurons to perform computations was first step toward computation and learning. Marvin Minsky and Dann Edmonds (1951) constructed the first neural network computer. In 1950: Alan Turning's "Computing Machinery and Intelligence" was the first complete vision of AI.

i) The birth of AI (1956): Dartmouth workshop bringing together top minds on automata, neural nets and the study of intelligence organized a two-month workshop at Dartmouth in the summer of 1956.

ii) Great expectations (1952-1969):

- → Newell and Simon introduced the general problem solver which was imitation of human problem solving.
- → John Mc Carthy (1958) was the inventer of Lisp (second-oldest high-level language).
- → Marvin Minsky (1958) introduced microworlds that appear to require intelligence to solve, anti-logic orientation and society of mind.

iii) Collapse in AI research (1966-1973):

- → Progress was slower than expected.
- → ~~Unrel~~ Unrealistic predictions.
- → Some systems lacked scalability.
- → Fundamental limitations on techniques and representations.

iv) AI revival through knowledge-based systems (1969–1970).

→ General purpose vs. domain specific.
→ Expert systems.
→ Increase in knowledge representation research.

v) AI becomes an industry (1980–present):

→ The first successful commercial expert system R1 began operation at the Digital Equipment Corporation (Mc Dermott, 1982).
→ Nearly every major U.S. corporation had its own AI group and was either using or investing expert systems saving millions of dollar per year.
→ The AI industry boomed from a few million dollars in 1988, including hundreds of companies building expert systems, vision systems, robots etc.

vi) The return of neural networks (1986–Present):

→ Back-propagation learning algorithm was applied that resulted Parallel Distributed Processing (Rumelhart and Mc Clelland, 1986) caused great excitment.
→ Seperation of AI and cognitive science in two fields, one concerned with creating effective network architectures and algorithms and understanding their mathematical properties, the other concerned with careful modeling of the empirical properties of actual neurons.

vii) AI becomes a science (1987–Present):

→ In speech recognization.
→ In neural networks.
→ In uncertain reasoning and expert systems.

viii) The emergence of intelligent agents (1995–Present):

→ One of the most important environments for intelligent agents is the Internet.
→ AI systems have become so common in web-based applications.
→ AI technologies underlie many internet tools, such as search engines, recommender systems and website aggregators.

Foundations of AI:

Foundations are the disciplines that contributed ideas, viewpoints and techniques to AI. Following are some of the foundations of AI.

i) **Philosophy:** It includes logic, methods of reasoning, mind as physical system, foundations of learning language, rationality etc. It leads to following type of questions:
   - → Where does knowledge come from?
   - → How does knowledge lead to action?
   - → Can formal rules be used to draw valid conclusions?

ii) **Mathematics:** It includes formal representation and proof algorithms, computation, probability etc. It leads to following type of questions:
   - → What can be computed?
   - → How do we reason with uncertain information?
   - → What are the formal rules to draw valid conclusions?

iii) **Psychology:** It includes adaption, phenomena of perception and motor control. It leads to following question.
   - → How humans and animals think and act?

iv) **Economics:** It includes formal theory of rational decisions, game theory, operation research etc. It leads to following questions:
   - → How should we make decisions so as to maximize pay off?
   - → How should we do this when the pay off may be far in future?

v) **Linguistics:** It includes knowledge representation and grammer. It leads to question
   - → How does language relate to thought?

vi) **Neuroscience:** It includes physical substrate for mental activities.
   - → How do brains process information?

vii) **Control theory:** It includes homeostatic systems, stability, optimal agent ~~decision~~ design.
   - → How can artifacts operate under their own control?

⊛Applications of AI:

AI is making our daily life more comfortable and fast because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education etc. Following are some sectors which have applications of AI.

i) AI in Healthcare: Healthcare industries are applying AI to make better and faster diagnosis than humans. AI can help doctors with diagnosis and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

ii) AI in Finance: AI and finance industries are the best matches for eachother. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading and machine learning into financial processes.

iii) AI in Gaming: AI can be used for gaming purposes. The AI machines can play strategic games like chess, where the machine needs to think of large number of possible places.

iv) AI in Social Media: Social Medias such as facebook, twitter, and snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amount of data. AI can analyze lots of data to identify the latest trends, hashtag and requirement of different users.

v) AI in data security: AI can be used to make data more safe and secure since cyber attacks are rapidly growing in the digital world. Some examples such as AEG bot, AI2 platform are used to determine software bugs and cyber attacks in a better way.

vi) AI in education: AI chatbot can communicate with students as a teaching assistant. AI in the future can work as a personal virtual tutor for students which will be accessible easily at any time and any place.

vii) AI in entertainment: We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recomendations for programs or shows.

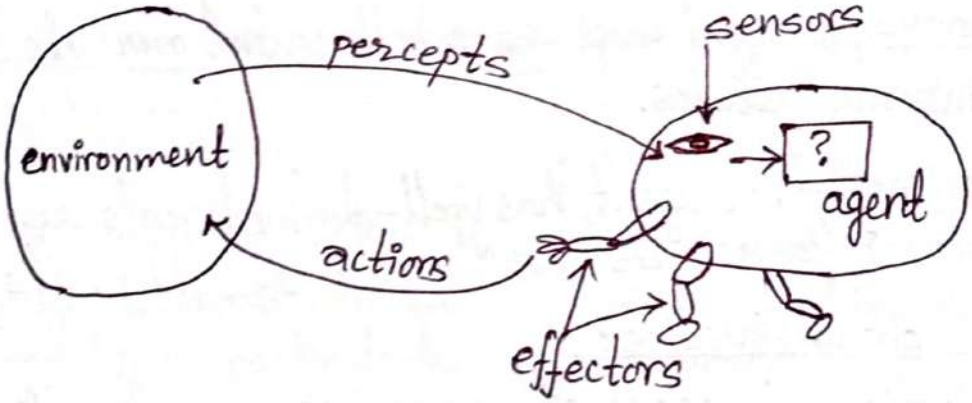
Unit-2
Intelligent Agents

⊛ Introduction:

An agent is anything that can be aware of its environment through sensors and acting upon that environment through actuators. Actuator is a device that causes machine or other devices to operate. An gent gets percepts one at a time and maps this percept sequence to actions.

Percepts → Percepts are the electrical signals from sensors after processing objects in visual field. (like location, colors, loudness, direction etc.)

Structure of intelligent agent:



Agent = Architecture + Program

Architecture is the machinery that the agent executes on. It is a device with sensors and actuators. For example a robotic car, camera, PC. Agent program is an implementation of an agent function. An agent function is a map from percept sequence (history of all that an agent has perceived till date) to an action.

Examples of agent:

i) A software agent → It has keystrokes, file contents, network packages which act as sensors and displays on the screen. Files, sent network packets acting as actuators.

ii) A Human agent → It has eyes, ears, and other organs which act as sensors and hand, legs, mouth etc acting as actuators.

iii) A Robotic agent → It has cameras and infrared range finders which act as sensors and various motors acting as actuators.

⊛ Properties of Intelligent Agents:

ⓐ Internal characteristics:

i) Learning → An agent has ability to learn from previous experience and to successively adapt its own behaviour to the environment.

ii) Reactivity → An agent must be capable of reacting appropriately to information from its environment.

iii) Autonomy → An agent must have both control over its actions and internal states.

iv) Goal-oriented → An agent has well-defined goals and gradually influence its environment to achieve its goals.

ⓑ External characteristics:

i) Communication → An agent often requires an interaction with its environment to fulfill its tasks, such as human and other agents.

ii) Cooperation → Cooperation of several agents provides better and faster solutions for complex tasks.

iii) Mobility → An agent may navigate with electronic communication networks.

iv) Character → Like human, an agent may demonstrate an external behaviour with many human characters as possible.

Rational Agent:

For each possible percept sequence, a rational agent is that which is expected to maximize its performance measure, given the evidence provided by the precept sequence and whatever built-in knowledge the agent has.

In short a rational agent is one that does right thing. Right thing or action is the one that will cause the agent to be most successful. At any given time rational agent depends on following four things:

i) The performance measure that defines the criteria of sucess.
ii) The agent's prior knowledge of the environment.
iii) The actions that the agent can perform.
iv) The agents percept sequence to date.

Differences between AI and Omniscience:

AI	Omniscience
i) AI is simulation of human intelligence demonstrated by machines, particularly computer systems.	i) Omniscience refers to the capacity of knowing unlimited knowledge of all things that can be known.
ii) The idea is to get the machines to think for themselves and make decisions based on data being fed.	ii) Omniscience is the state of possessing unlimited or complete knowledge about all things possible.
iii) AI is based on algorithms created by humans to help the machines think and learn.	iii) It is an attribute given to the god alone because in reality, omniscience is impossible.

⊛. Configuration of Agent:

To design a rational agent we must specify its task environment. Task environment means: PEAS description of the environment.

P → Performance
E → Environment
A → Actuators
S → Sensors.

⊛. PEAS description of Agents:-

For PEAS description of agents lets take an example of fully automated taxi. Now, the agent type becomes taxi driver and PEAS description of task environment for an automated taxi is as follows:-

Performance → Safe, fast, legal, comfortable, maximize profits.

Environment → Roads, traffic signals, weather.

Actuators → Steering, accelerator, brakes, horn.

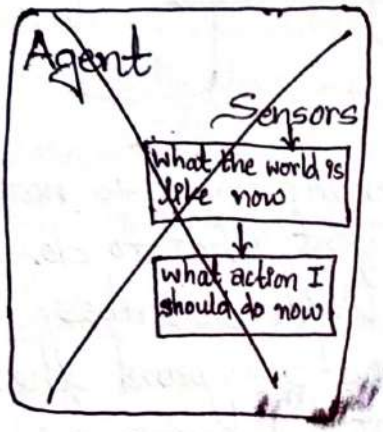
Sensors → Cameras, sonar, speedometer, GPS, odometer, engine sensors.

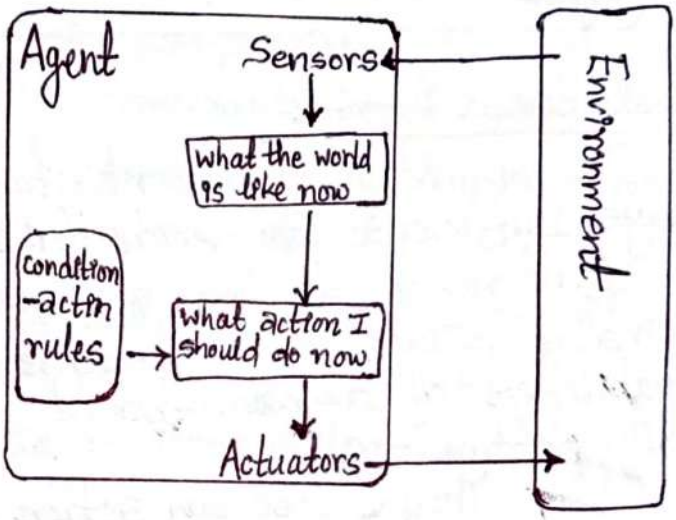
⊛. Types of Agents:-

1. Simple Reflex Agent: The simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history. These agents only succeed in the fully observable environment.

The simple reflex agent works on condition-action rule, which means it maps the current state to action. ~~Such as~~ i.e, if the condition is true, then the action is taken, else not. Such as a Room Cleaner agent, it works only if there is dirt in the room.

The problem with simple reflex agent is that they have very limited intelligence and they are not adaptive to changes in the environment.





2. Model-Based Agents:-

The model-based agent can work in a partially observable environment, and track the situation. A model-based agent has two important factors:

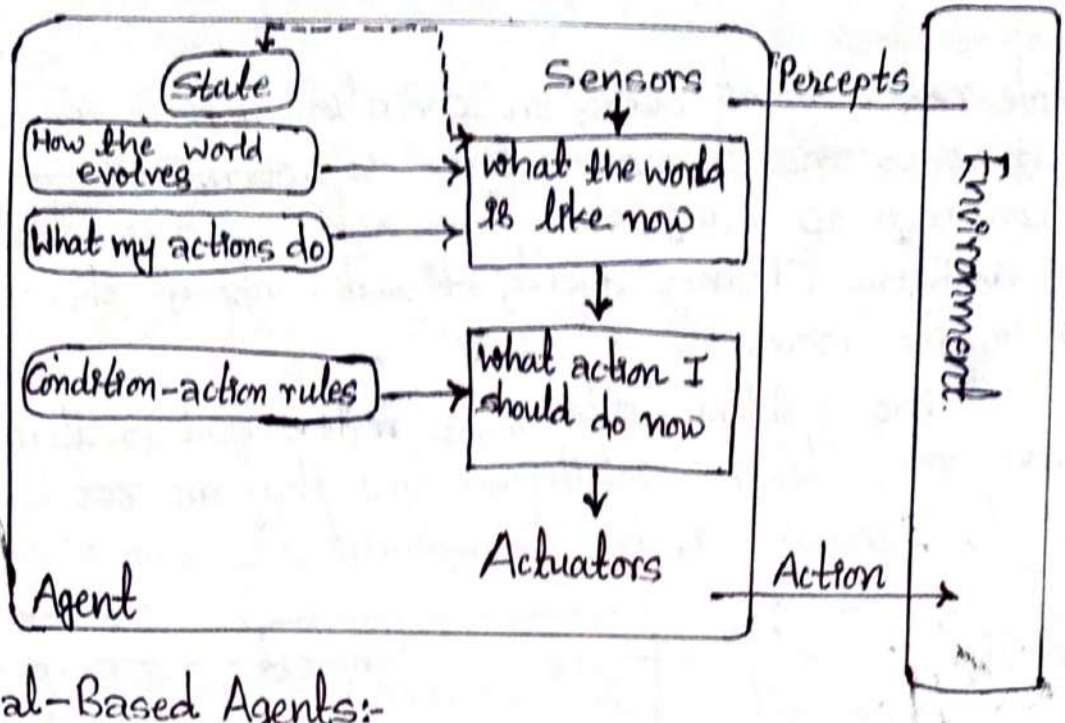
1) **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.

2) **Internal State:** It is a representation of the current state based on percept history.

These agents have the model, "which is knowledge of the world" and based on the model they perform actions. Updating the agent state requires information about:

→ How the world evolves.

→ How the agent's action affects the world.



3. Goal-Based Agents:-

The knowledge of current state environment is not always sufficient to decide for an agent what to do. The agent needs to know its goals which describes desirable situations. Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.

They choose an action, so that they can achieve the goal. These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.

> Note: Figure is same as above figure just replace condition-action rules by Goals and some small changes only have a look once.

4. Utility-based agents:

These agents are similar to the goal based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state. Utility-based agent act based not only goals also the best way to achieve goal.

The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action. The utility function

maps each state to a real number to check how efficiently each action achieves the goals.

Note:- Figure is similar but small certain chages so look once.

5. Learning Agents:

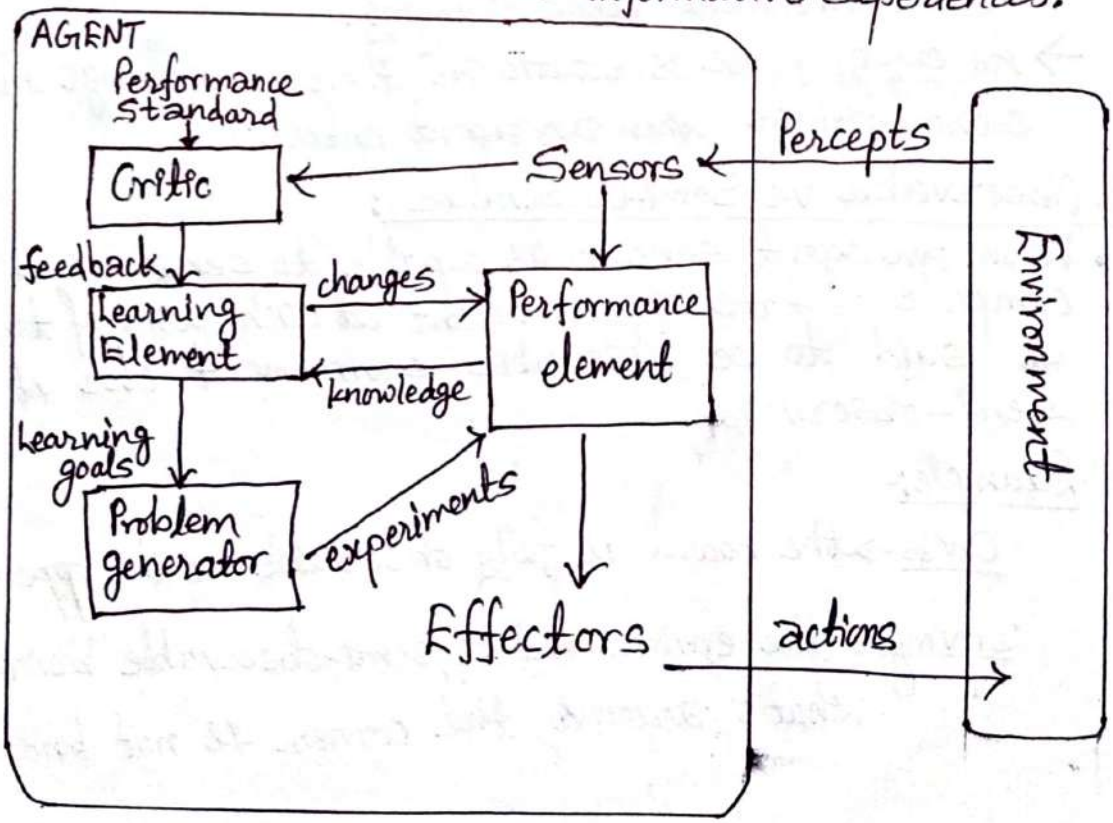
A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities. It starts to act with basic knowledge and then able to act and adopt automatically through learning. Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance. A learning agent has mainly four conceptual components, which are:-

i) Learning element → It is responsible for making improvements by learning from environment.

ii) Critic → It describes how well that agent is doing with respect to a fixed performance standard.

iii) Performance element → It is responsible for selecting external action.

iv) Problem generator → It is responsible for suggesting actions that will lead to new and informative experiences.



⊛. Environment Types:

1. Deterministic vs Stochastic:

- When an uniqueness in the agent's current state completely determines the next state of the agent, the environment is said to be deterministic.
- Stochastic environment is random in nature which is not unique and cannot be completely determined by the agent.

Example:

Chess → there would be only few possible moves for a coin at the current state and these moves can be determined.

Self Driving Cars → the actions are not unique, it varies time to time.

2. Dynamic vs Static:

- An environment that keeps constantly changing itself when the agent is up with some action is said to be dynamic.
- An idle environment with no change in its state is called a static environment.

Example:

→ A roller coaster is dynamic as it is set to motion and the environment keeps changing.

→ An empty house is static as there's no change in the surroundings when an agent enters.

3. Observable vs Semi-Observable:

- When an agent sensor is capable to sense or access the complete state of an agent at each point of time, it is said to be observable environment else it is semi-observable.

Example:-

Chess → the board is fully observable, so the opponent moves.

Driving → the environment is semi-observable because what is around the corner is not known.

4. Single-agent vs Multi-agent:

- An environment consisting of only one agent is said to be a single agent environment.
- An environment involving more than one agent is a multi agent environment.

Example:

→ A person left alone in a maze is an example of single agent system.

→ The game of football is multi agent as it involves 10 players in each team.

5. Discrete vs Continious:

- The environment in which the actions performed ~~cannot~~ can be numbered is said to be discrete environment.
- The environment in which the actions performed cannot be numbered is said to be continious environment.

Example:

→ The game of chess is discrete as it has only finite number of moves. The number of moves might vary with every game, but still, it's finite.

→ Self-driving cars are continious as their actions like driving, parking etc. cannot be numbered.

UNIT-3

Problem Solving By Searching

Problem solving in AI is a systematic search through a range of possible action in order to reach some predefined goal or solution. There are four general states in problem solving which are as follows:

i) Goal ~~formation~~ formulation:
A goal is a state that the agent is trying to reach.

ii) Problem formulation:
This is the process of deciding what actions and states to consider, given goal.

iii) Search Method:
It determines possible sequence of actions and choose the best one.

iv) Execute:
It provides the solution to perform action.

Searching → It is a commonly used method in AI for solving problems. The search technique explores the possible moves that one can make in a space of 'states', called the search space.

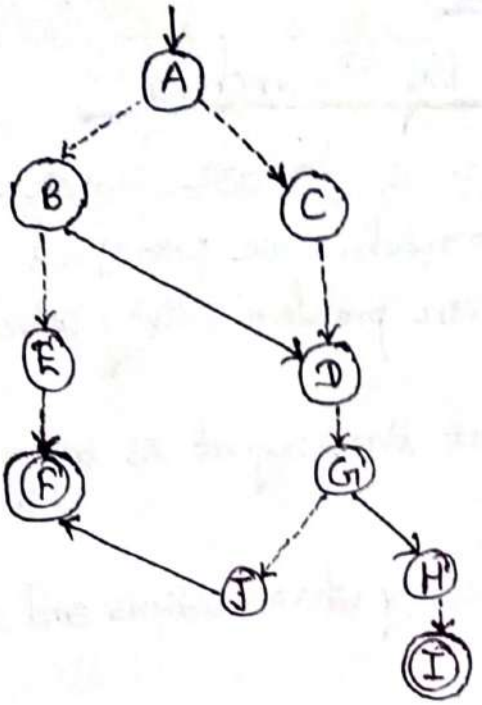
Problem Formulation → It involves deciding what actions and states to consider, given the goal. A problem can be defined formally by 4 components as;

i) An initial state: from which agent start.
ii) State Description: Description of possible action available on agent. It is also called successor function.
iii) Goal test: Determining the state is goal state or not.
iv) Path cost: Sum of cost of each path from initial to given state.

State Space Representation → The state space is defined as a directed graph with each node is a state and arc represents the application of an operator transforming a state to a successor state. A solution is path from the initial state to goal state consisting of (S, s, O, G).

where, S = Set of states
s = start state
O = Operation (which helps to transfer state to its successor state).
G = Goal state.

Example:-



$S = \{A, B, C, D, E, F, G, H, I, J\}$

$s = A$

$G = \{F, I\}$

Well-Defined Problem: A problem that lacks one or more of the below 5 properties is an ill-defined problem. But a well-defined problem can be defined formally by following five components:

i) Initial state → State from which agent start.

ii) State Description (Successor function) → Description of possible actions available on agent.

iii) State Space and Path → All states reachable from initial by any sequence of actions is state space. Path is the sequence through state space.

iv) Path cost → Sum of cost of each path from initial to given state.

v) Goal test → Determining the state is goal state or not.

Well defined problem
- → Vaccum world state problem.
- → Real world Problem (Used to define the time, money and human resource cost).
- → Toy problem (Used to compare performance of system).

⊛. Solving Problems by Searching: (Concept) ← less imp

Figure below contains a representation of a map. The nodes represent cities, and the links represent direct road connections between cities. The number associated to a link represents the length of the corresponding road. The search problem is to find a path from a city S to a city G.

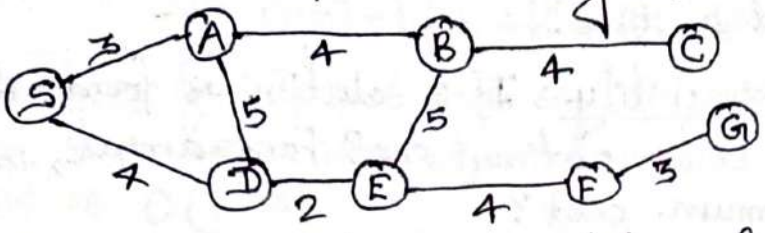


Fig. A graph representation of map.

This problem will be used to illustrate some search methods. Search problems are part of a large number of real ~~word~~ world applications:

- → VLSI Layout
- → Web Search
- → Path Planning
- → Robot navigation etc.

⊛. Search Techniques / Search Strategies:

There are two broad ~~sear~~ search strategies:

1) Uninformed (or blind) search methods → In the method, the order in which potential solution paths are considered is arbitrary, using - no domain specific information to judge where the solution is likely to lie. E.g. Breadth first search, Depth first search, Depth limit search, Bidirectional search etc.

2) Heuristically informed search methods → In this method, one uses domain-dependent (heuristic) information in order to search the space more efficiently. E.g. Greedy best first search, A* search, Hill climbing search etc.

⊛ Performance Evaluation of Search Techniques:-

We can evaluate the performance of search techniques in four ways:

1) Completeness → An algorithm is said to be complete if it definitely finds solution to the problem, if exist.

iii) Time Complexity → How long (worst or average case) does it take to find a solution? Usually measured in terms of number of nodes expanded.

iv) Space Complexity → How much space is used by the algorithm? Usually measured in terms of the maximum number of nodes in memory at a time.

v) Optimality / Admissibility → If a solution is found, is it guaranteed to be an optimal one? For example, is it the one with minimum cost?

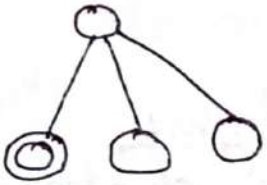
⇒ Time and Space complexity are measured in terms of:

⇒ b → Maximum branching factor (number of successor of any node) of the search tree.

⇒ m → Depth of the least-cost solution.

⇒ d → Maximum length of any path in the space.

Example:



b = 3
d = 1
m = 1

1. Uniformed Search:

a. Breadth First Search (BFS): It is an algorithm for traversing or searching tree or graph data structures. It starts with the tree root and explores all of the neighbour nodes at the present depth prior to moving on to the nodes at the next depth level. It is implemented in FIFO queue in DSA.

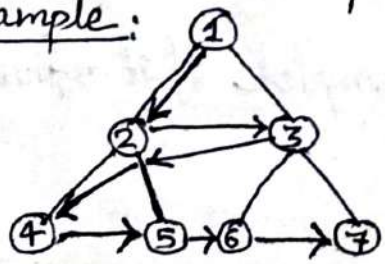
In BFS the graph is traversed as follows:

→ First move horizontally and visit all the nodes of the current layer.

→ Move to the next layer.

→ Do not generate as child node if the node is already parent to avoid more loop. (i.e, avoid loop formation).

Example:



The traversing order of given graph in BFS is 1,2,3,4,5,6,7. (as shown by arrow in figure).

Performance Measures of BFS

Completeness → It always finds the solution if exist. If shallowest goal node is at some finite depth d and if b is finite.

Time complexity → Assume a state space where every state has b successors, then

$$O(n) = b + b^2 + b^3 + \ldots + b^d = O(b^d).$$

Space complexity → If goal node is at depth d then the nodes of d depth are expanded before traversing so space complexity of BFS is $O(b^{d+1})$.

Optimal → If all the path has same cost then optimal otherwise not.

Major lessons from BFS are:

→ Memory management is the biggest issue than it's execution time.
→ Exponential complexity can not be computed in uninformed search.

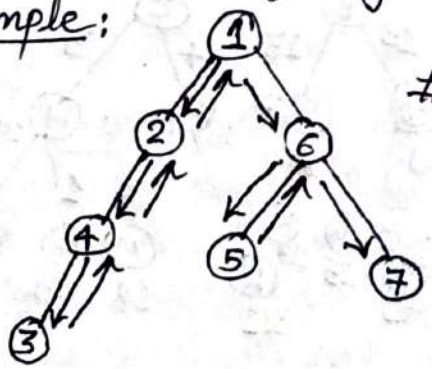
Applications:

→ Social Networking Websites use BFS.
→ In GPS navigation systems.
→ Broadcasting in network.

(b) Depth First Search (DFS): It is another method for traversing or searching tree or graph data structures. This algorith starts at root node and explores as far as possible along each branch before backtracking. It is implemented in LIFO architecture (stack).

The main strategy of DFS is to explore deeper into the graph whenever possible. When all the edges have been explored, the search backtracks until it reaches an unexplored neighbor. This process continues until all the vertices are reached without forming any loop.

Example:



Traversal of DFS is 1,2,4,3,6,5,7 as shown by arrows in graph.

Performance Measure/Evaluation of DFS:

Completeness → It does not find the solution in all cases (if depth is infinite and DFS contains loops).

Time Complexity → Let m be the maximum depth, then time complexity will be

$$b+b^2+b^3+\ldots+b^m$$
$$= O(b^m).$$

Space Complexity → Need to store single node so;
$$= (b+b+b+\ldots) * m \text{ time}$$
$$= O(bm).$$

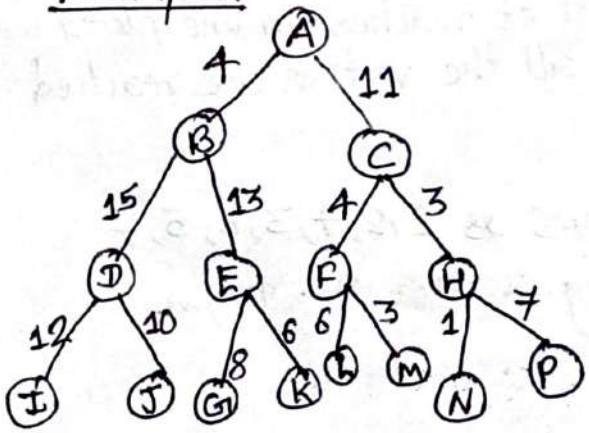
Optimality → It gives optimal solution than BFS but it does not give optimal solution if the goal node is in right child of level 2 or 3 of graph having m=10 or more.

Applications:
- → Detecting cycle in a graph.
- → Path Finding.
- → Topological sorting
- → Solving puzzle with only one solution, such as mazes.

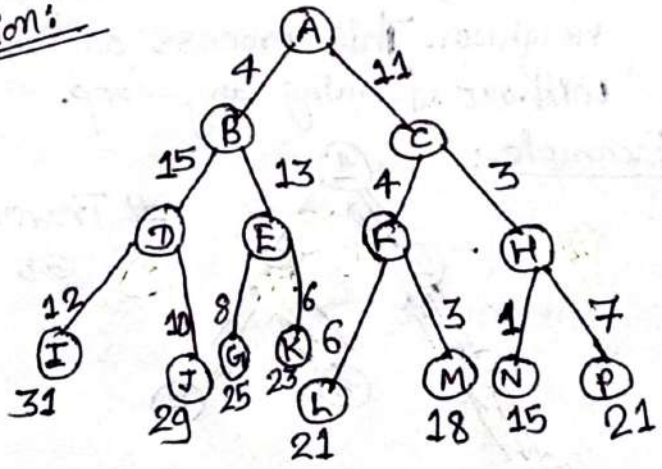
© Uniform Cost Search (UCS):-

It is the modified version of BFS to make optimal. This algorithm is used for traversing or searching a weighted tree or graph. The search begins at the root node. The search continues by visiting the next node which has the least total cost from the root. It does not care about the number of steps, only care about the total cost.

Example:



Solution:



Performance measure of UCS:

Completeness → Yes it satisfies this property, because cost can be definitely calculated for graph with finite depth.

Space Complexity → Maximum of BFS.

~~Time~~ Time Complexity → Maximum of BFS.

Optimal → Yes, because at every step the path with the least cost is chosen.

Conclusion ⇒ UCS can be used instead of BFS in case that path cost is not equal and is guaranteed to be greater than a small positive value e.

(d). Depth Limited Search (DLS):

It is the modification of depth-first search. It is an algorithm to explore the vertices of graph. It works exactly like depth-first search but avoids the drawbacks regarding completeness by imposing a maximum limit on the depth of the search. It solves the infinite-path problem of DFS.

Algorithm:

```
depth limit = max depth to search to;
agenda = initial stage;
If initial stage is goal stage then return solution
else
  while agenda not empty do
      take node from front of agenda;
      if depth(node) < depth limit then
      {
        new does = apply operations to node;
        add new nodes to front of agenda;
        if goal stage in new nodes then return solution;
      }
```

Performance measure of DLS

Completeness → In case limit $l < d$, DLS will ~~not~~ never reach a goal, so in this case ~~incomplete but~~ DLS is not complete.

Time complexity → $O(b^l)$

Space complexity → $O(b^l)$

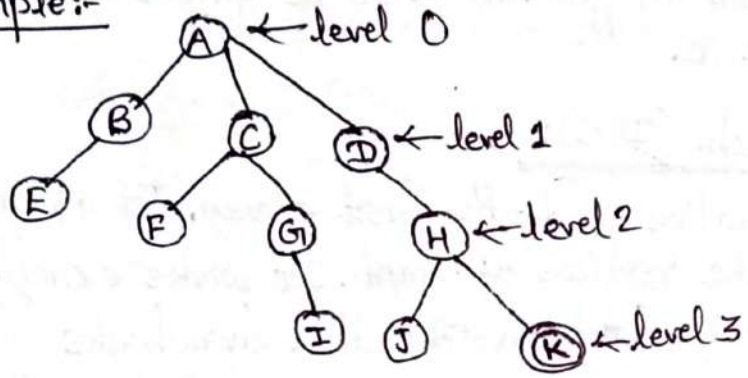
Optimality → If DLS is complete then it is optimal than DFS.

(e) Iterative Deeping First Search:

This technique is the combination of BFS and DFS so that the completeness and optimality of BFS and the memory requirement level of DFS could be achieved.

In this strategy, depth-limited search is run repeatedly, increasing the depth limit with each iteration until it reaches d, the depth of the shallowest goal state. On each iteration this method visits the nodes in the search tree in the same order as depth-first search, but the cumulative order in which nodes are first visited.

Example:-



Depth level	IDFS	Goal
0	A	No
1	ABCD	No
2	ABCDEFGH	No
3	ABCDEFGHIJK	Yes.

Performance Measure of IDFS:

Completeness → IDFS is like BFS, It is complete when the branching factor b is finite.

Space Complexity → $O(b^{d/2})$

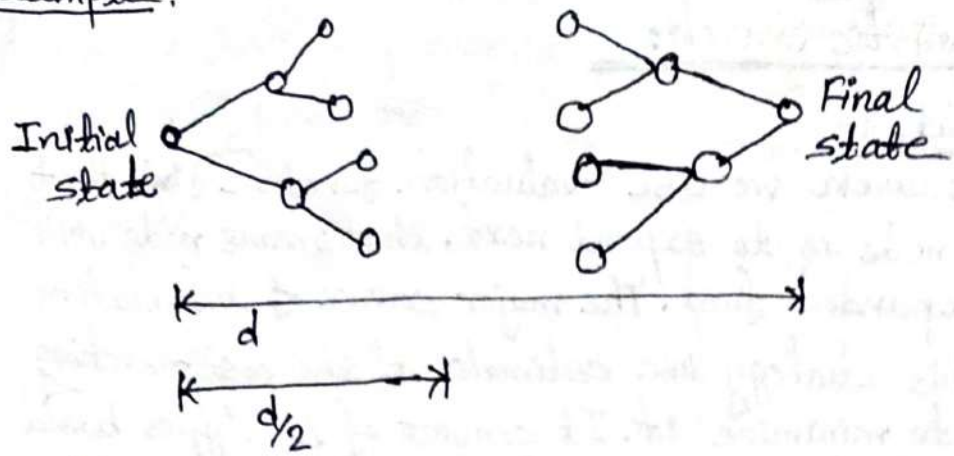
Time Complexity → $O(b^{d/2})$.

Optimality → IDFS is also like BFS, so optimal when the steps are of same cost.

(f). Bidirectional Search:

Bidirectional search suggests to run two simultaneous search graph, one from the initial state and the other from the goal state. It then expands node from the start and goal state simultaneously. Check at each stage if they meet in the middle. If so, the path concatenation is the solution.

Example:



Performance measure of Bidirectional search:

Completeness → Yes it is complete when we use BFS in both searches.

Time Complexity → $O(b^{d/2})$.

Space Complexity → $O(b^{d/2})$

Optimality → Optimal when BFS is used and paths are of a uniform cost.

⊛ Comparision of All Uniformed Searches:

Strategy	Complete	Optimal	Time Complexity	Space Complexity
BFS	Yes	Yes	$O(b^d)$	$O(b^{d+1})$
DFS	No	No	$O(b^d)$	$O(bm)$
DLS	Yes(if $l>d$)	No	$O(b^m)$	$O(bm)$
IDFS	Yes	Yes	$O(b^d)$	$O(b^d)$
BDS	Yes	Yes	$O(b^{d/2})$	$O(b^{d/2})$

→ BFS and BDS are complete and optimal but consumes more space

→ DFS require less memory if the maximum tree depth is limited but cannot guarantee the solution.

→ DLS offers an improvement over DFS.

→ Best algorithm is IDFS or IDS, which is complete, optimal less space consumption but it cannot give solution in exponential time.

⊛. Heuristic function $h(n)$:-

Heuristic function $h(n)$ estimates the "goodness" of a node n. It is an estimate based on domain-specific information that is computable from current state description, of how close we are to a goal. $h(n)$ = estimated cost (or distance) of minimal cost path from n to a goal state.

2) Informed or Heuristic Search:

(a) Best-First Search :-

In this search we use evaluation function $f(n)$ that determines which node is to expand next, that means node with lowest $f(n)$ is expanded first. The major source of information for $f(n)$ is $h(n)$. This strategy use estimates of the cost reaching the goal and try to minimize it. It consist of two types based on the evaluation function with special cases which are as follows:-

1) Greedy Best-First Search:

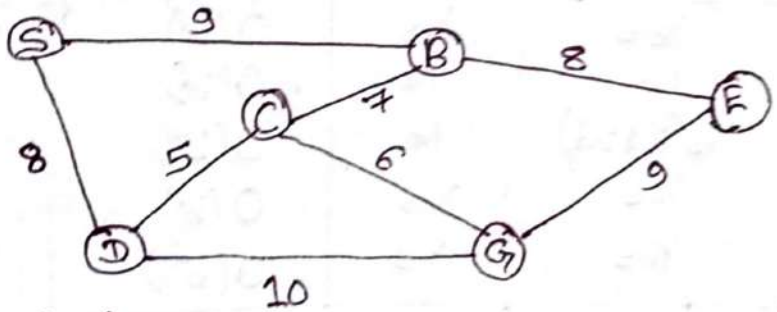
This method uses an evaluation function to select which node is to be expanded next. The node with the lowest evaluation is selected for expansion because it has the closest path to the goal.

Evaluation function $f(n) = h(n)$

$= $ ~~or~~ estimate of cost from n to goal.

This algorithm has $O(b^m)$ space and time complexity. Where b is the average branching factor and m is the maximum depth of the search tree.

Example:



Let G1 be goal node, then distances to node G1 from other node are:

$S \rightarrow G1 = 12$ ← given values

$B \rightarrow G1 = 4$

$E \rightarrow G1 = 7$

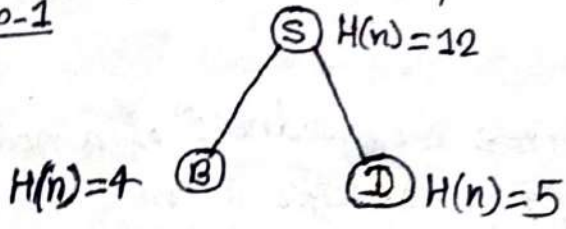
$D \rightarrow G1 = 5$

$C \rightarrow G1 = 3$

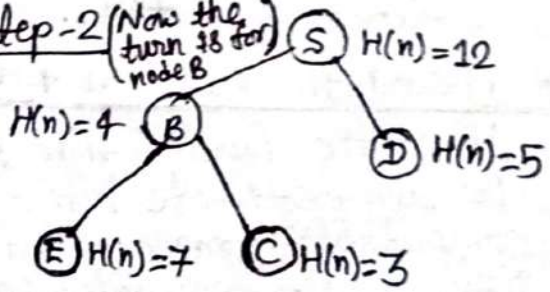
Solution:

Now greedy search operation is done as below.

Step-1

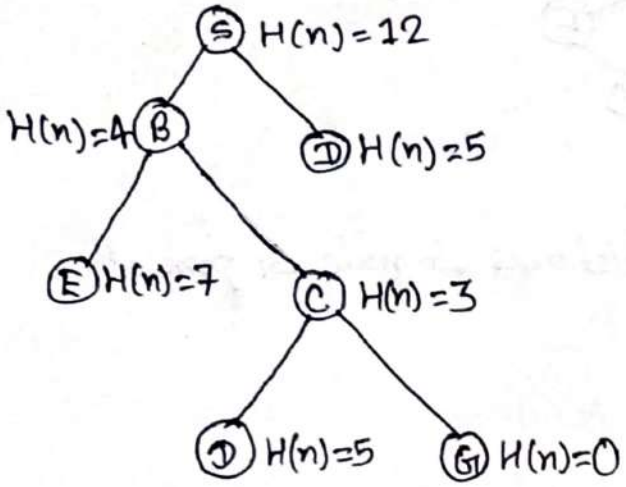


Step-2 (Now the turn is for node B)



Step:-3

Now the turn is of node C



Step:-4

Now the turn is of node G1, but we reached our goal node so we stop here.

Performance Measures:

Completeness → It is not complete because while minimizing the value of heuristic greedy may oscillate between two nodes.

Time complexity → $O(b^m)$.

Space complexity → $O(b^m)$.

Optimality → It is not optimal.

A* Search:

Admissible Heuristic → A heuristic function is said to be admissible if it is no more than the lowest-cost path to the goal. It is used to estimate the cost of reaching the goal state in an informed search algorithm. The estimated cost must always be lower than the actual cost of reaching the goal state.

A* search uses heuristic function $h(n)$ as well as the path cost to the node $g(n)$ in computing the total cost $f(n)$.

$$f(n) = h(n) + g(n)$$

where, ~~h(n) within the f(n) it must be admissible heuristic~~.

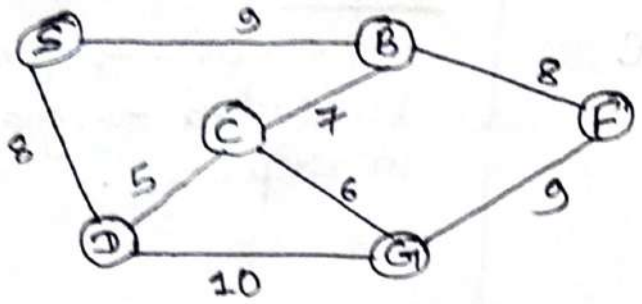
$f(n)$ = the evaluation function.

$h(n)$ = estimated cost from current node to goal.

$g(n)$ = the cost from the start node to current node.

This method follows a path of the lowest known path keeping a sorted priority queue of alternate path segments along the way. If path being traversed has higher cost than another path then it neglects the higher cost path and traverses lower cost path segment. This process continues until goal is reached.

Example:

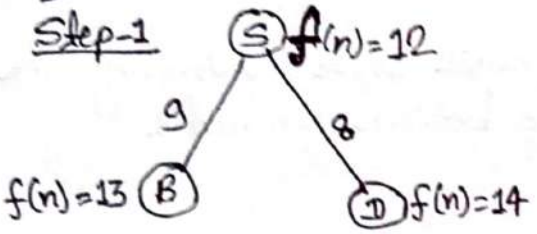


Let G be the goal node, then distances to node G from other node are:

$S \rightarrow G = 12$
$B \rightarrow G = 4$
$E \rightarrow G = 7$
$D \rightarrow G = 6$
$C \rightarrow G = 3$

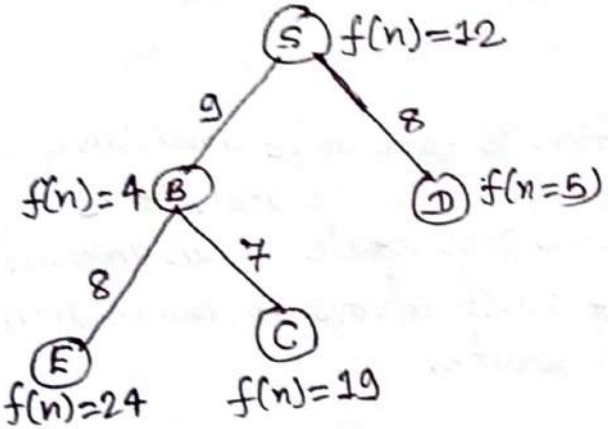
Solution:

Step-1



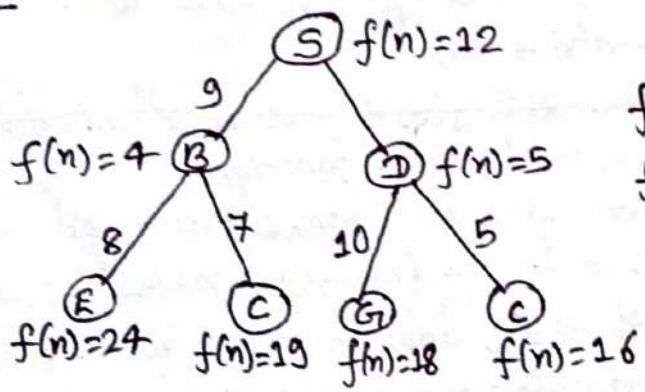
$f(B) = 9+4 = 13$
$f(D) = 8+6 = 14$

Step-2: Now the turn is of node B.



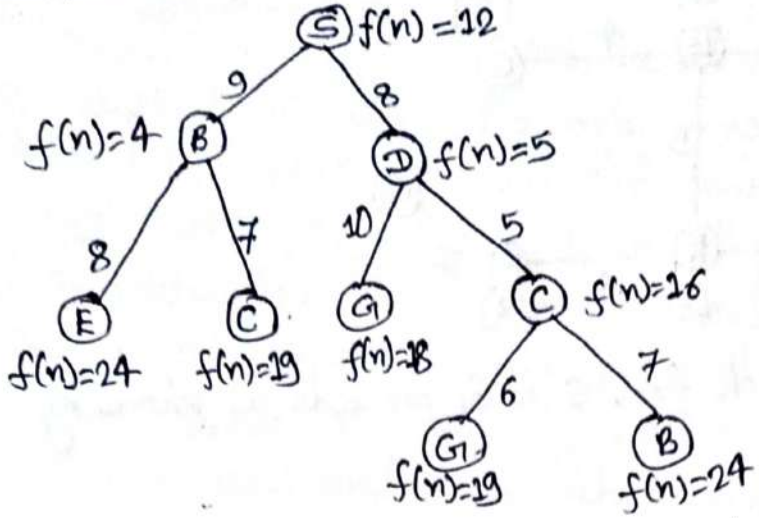
$f(E) = 17+7 = 24$
$f(C) = 16+3 = 19.$

Step-3: Now the turn is of node D.



$f(G) = 18+0 = 18$
$f(C) = 13+3 = 16$

Step:-4 Now turn is of node C.



$f(G_1) = 19 + 0 = 19$
$f(C) = 20 + 4 = 24$

G_1 is goal node, hence we are done.

Evaluation / Performance Measures:-

Completeness → Yes A* seach always give us solution.

Time Complexity → $O(b^d)$.

Space Complexity → It keeps all generated nodes in memory. Hence space is the major problem not time.

Optimality → A* gives optimal solution.

Admissibility → A* is admissible and considers fewer nodes than any other admissible search algorithm with the same heuristic.

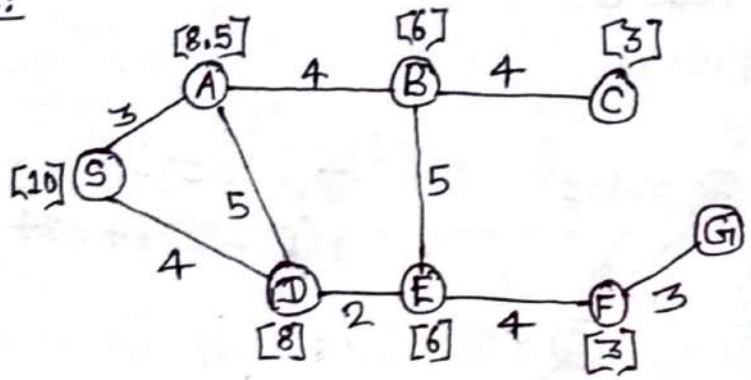
(b) Hill Climbing Search:-

This search technique can be used to solve problems that have many solutions, some of which are better than others. It starts with a random solution, and iteratively makes small changes to the solution, each time improving a little. When the algorithm cannot see any improvement anymore, it terminates. It can be described as follows:-

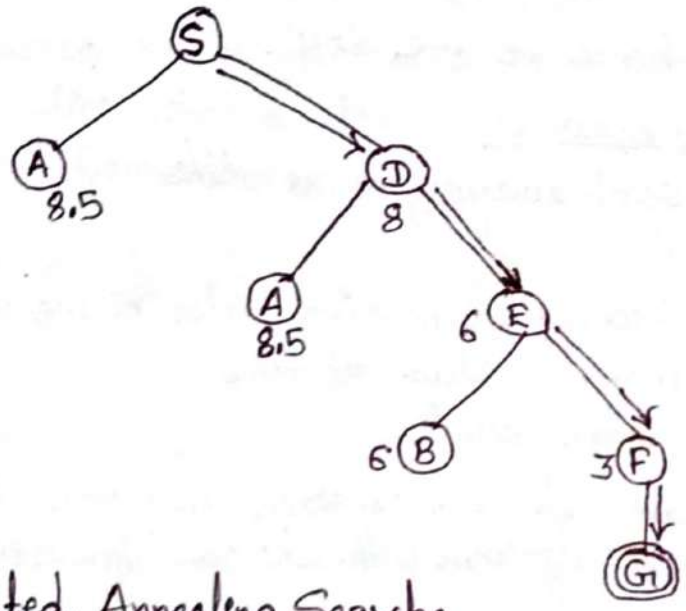
i) Evaluate the initial state, if it is goal state then terminate. Otherwise, current state is initial state.
ii) Select a new operator for this state and generate a new state.
iii) Evaluate the new state
   → If it is closer to goal, make it current state.
   → If it is no better, ignore.
iv) If the current state is goal state then terminate. Otherwise repeat from (ii).

Hill climbing terminates when there are no successors of the current state which are better than the current state itself.

Example:



The hill climbing search from S to G proceeds as shown by arrow as follows:



© Simulated Annealing Search:

Compared to hill climbing the main difference is that simulated annealing allows downwards steps. It also differs from hill climbing that a move is selected at random and decides whether to accept it. If the move is better than its current position then it will be accepted. If the move is worse then it will be accepted based on some probability. The probability of accepting a worse state is given by the equation;

$$P = \text{exponential}(-c/t) > r$$

where, c = the change in the evaluation function
t = the current value
r = a random number between 0 and 1.

⊛. Game Playing:

Game Playing is an important domain of AI. Games don't require much knowledge; The only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game. Both players try to win the game. So both of them try to make the best move possible at each turn. Searching techniques like BFS are not accurate for this as the branching factor is very high, so searching will take a lot of time. So, we need another search procedures that improves searching time. Games are a form of multi-agent environment and arise questions like what do other agents do and how they affect our success? Competitive multi-agent environments give rise to games.

⊛ Adversarial Search Techniques:

Adversarial search is also known as Minimax search used in calculating the best move in two player games where all the information is available, such as chess or tic tac toe. It consists of navigating through a tree which captures all the possible moves in the game, where each move is represented in terms of loss and gain for one of the players.

Game-adversary

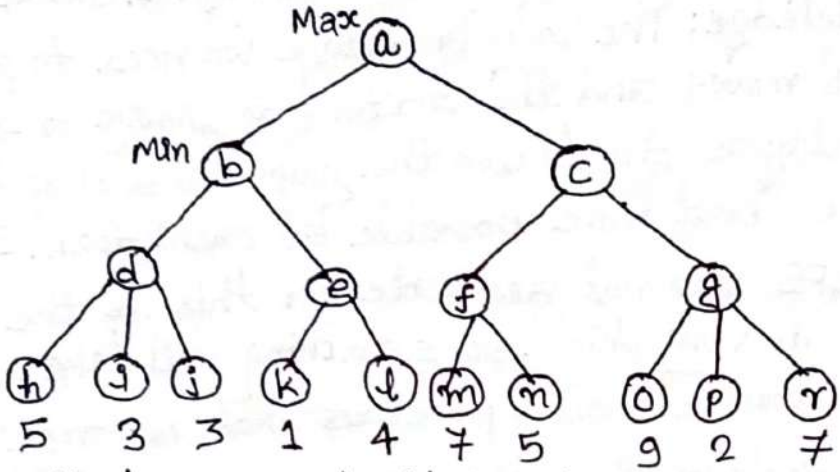
→ Solution is strategy (strategy specifies move for every possible opponent reply).
→ Time limits force an approximate solution.
→ Evaluation function: evaluate "goodness" of game position.
→ Examples: chess, checkers, backgammon.

1> The Mini-max Search:

Let us assign the following value for the game: 1 for win by X, 0 for draw by X, -1 for loss by X. Given the values of the terminal nodes (win for X(1), loss for X(-1), draw for X(0)), now the values of the non-terminal nodes are computed as follows:-

→ The value of a node where it is the turn of player X to move is the maximum of the values of its successors (because X tries to maximize its outcome).
→ The value of a node where it is the turn of player O to move is the minimum of the values of its successors (because O tries to minimize the outcome of X).

Example: Consider the following game tree (drawn from the point of view of the maximizing player):



Show that moves should be chosen by the two players, assuming that both are using mini-max procedure.

Solution:

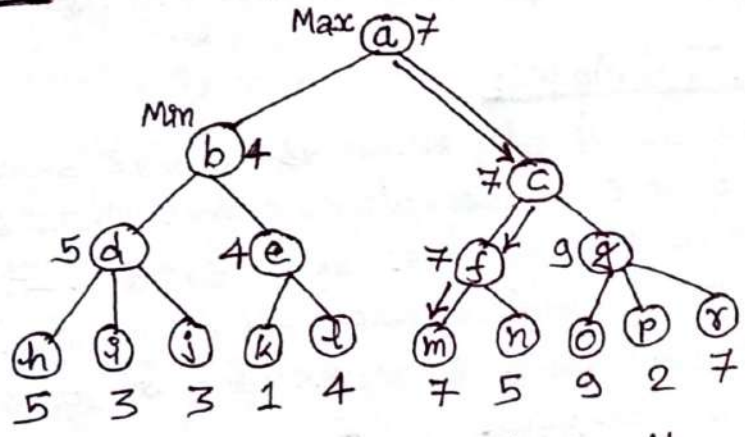


Fig. The min-max path for the game tree

2) Alpha-Beta Pruning:

This method is applied to a standard minmax tree, it returns the same move as minmax would, but prunes away branches that can not possibly influence the final decision. Alpha-Beta ~~Puring~~ Pruning can be applied to trees of any depth and it is often possible to prune entire sub-trees rather than just leaves. This is a technique for evaluating nodes of a game tree that eliminates unnecessary evaluations. It uses two parameters alpha and beta.

Alpha → It is the value of the best (i.e, highest value) choice we have found so far at any choice along the path for MAX.

Beta → It is the value of the best (i.e, lowest value) choice we have found so far at any choice along the path for MIN.

An alpha cutoff → Let us consider that the current board situation corresponds to the node A in the following figure:

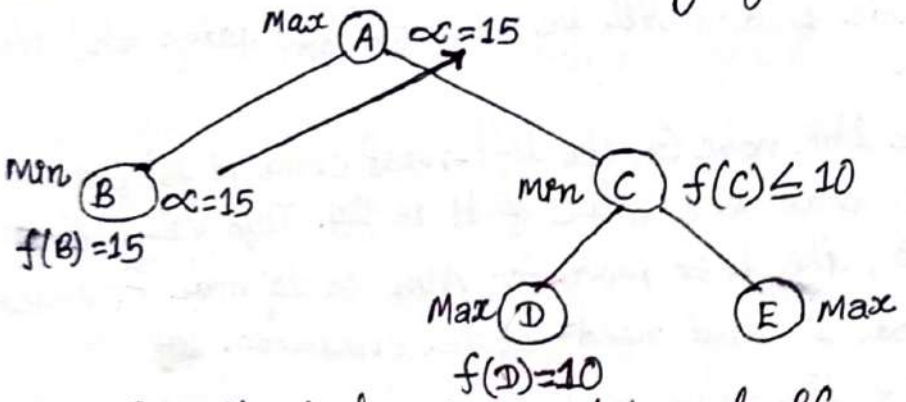


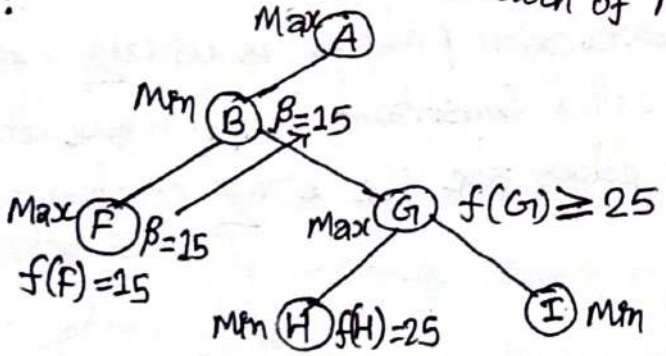
fig. Illustration of the alpha cut-off

The mini-max method uses a depth-first search strategy therefore it will estimate the value of node B first. Let node ~~be~~ B be evaluated 15. If Max will move to B then it is guaranteed to achieve 15. Therefore 15 is the lower bound for the achievement of max player. Therefore value of α at node B is 15 which is transmitted upward to node A and will be used for evaluating the other possible moves from A.

To evaluate the C, it's left-most child D has to be evaluated first. Let us assume that value of D is 10. This value is less than the value of α, the best move for MAX is ~~to~~ node B, independent of the value of node E that need not be evaluated.

If the value of E is greater than 10, Min will move to D which has the value 10 for Max, otherwise, Min will move to E which has a value less than 10. So, if Max moves to C, the best it can get is 10; which is less than the value $\alpha = 15$ that we get if Max would move to B. Therefore, the best move for Max is to B, independent of the value of E. The Elimination of node E is an alpha cutoff.

A beta cutoff → It represents an upper bound for the achievement of the Max player. In above tree, the Max player moved to node B. Now it is the turn of Min player to decide where to move!



Let us assume that the value of F has be evaluated to 15. Therefore the value of β at the node F is 15. This value is transmitted upward to node B and will be used for evaluating the other possible moves from B.

To evaluate the node G, it's left-most child H is evaluated first. let us assume that the value of H is 25. This value is greater than the value of β, the best move for Min is to node F, independent of value of node I that need not be evaluated.

If the value of I is greater or equal to 25, then Max(In G) will move to I otherwise Max will move to H. So in both cases the value obtained by Max is at least 25 which is greater than β. Therefore the best move for Min is at F, independent of the value of I. The elimination of the node I is a beta cutoff.

⊛. Constraint Satisfaction Problems: (With Examples):

A Constraint Satisfaction Problem (CSP) consists of a set of variables, a domain of values for each variable and a set of constraints. The objective is to assign a value for each variable such that all constraints are satisfied.
Examples: The n-Queen problem, Crossword puzzle, Sudoku etc.

A Constraint Satisfaction Problem is characterized by:
→ a set of variables $\{x1, x2, \ldots, xn\}$.
→ for each variable xi a domain Di with the possible values for that variable, and a set of constraints that are assumed to hold between the values of variables.

A Constraint Satisfaction Problem is to find, for each i from 1 to n, a value in Di for xi so that all constraints are satisfied.

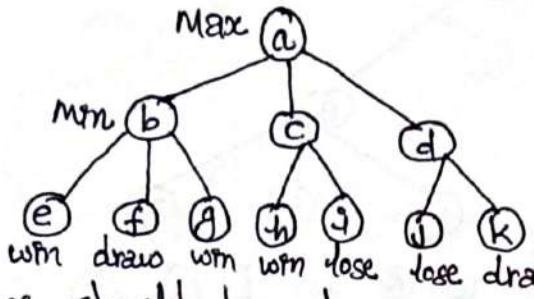
This problem can be easily stated as a sentence in first order logic of the form:

$$(\text{exist } 1) \ldots (\text{exist } xn)\ (D1(x1) \& \ldots Dn(xn)) \Rightarrow (C1 \ldots Cm)$$

A Constraint Satisfaction problem is usually represented as an undirected graph called Constraint Graph where the nodes are the variables and the edges are the binary constraints.

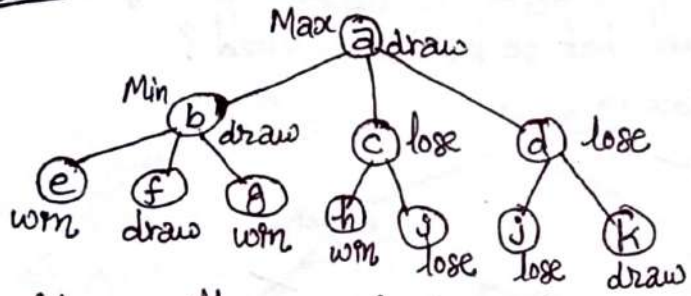
Examples: (Important)

Example 1: Consider the following game tree (drawn from the point of view of the Maximizing player):



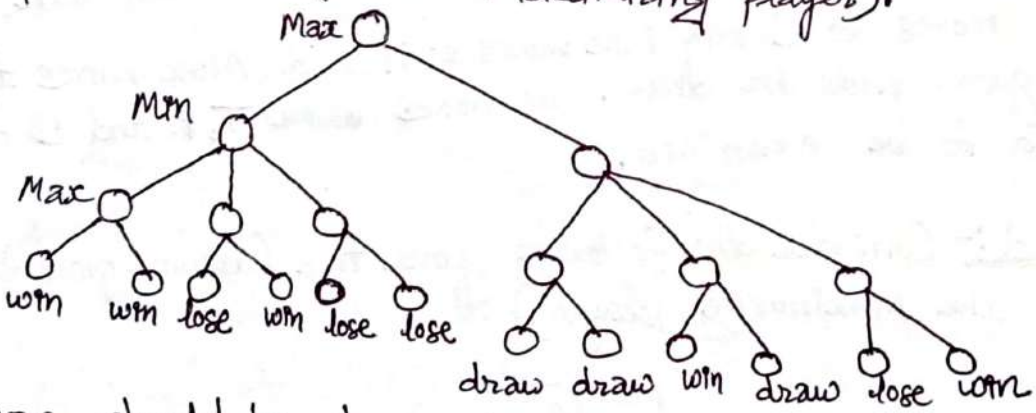
What move should be choosen by the Max player, and what should be the response of the Min player, assuming that both are using the mini-max procedure?

Solution:



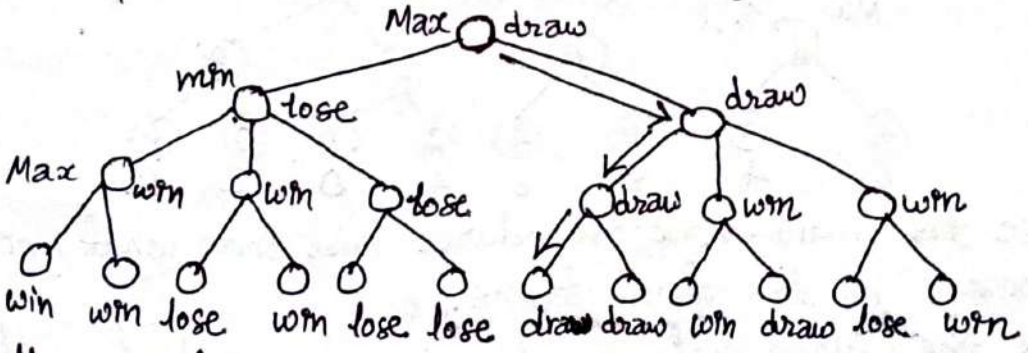
Max will move to b and min will respond by moving to f.

Example 2: Consider the following game tree (~~drawn~~ drawn from the point of view of the Maximizing player):



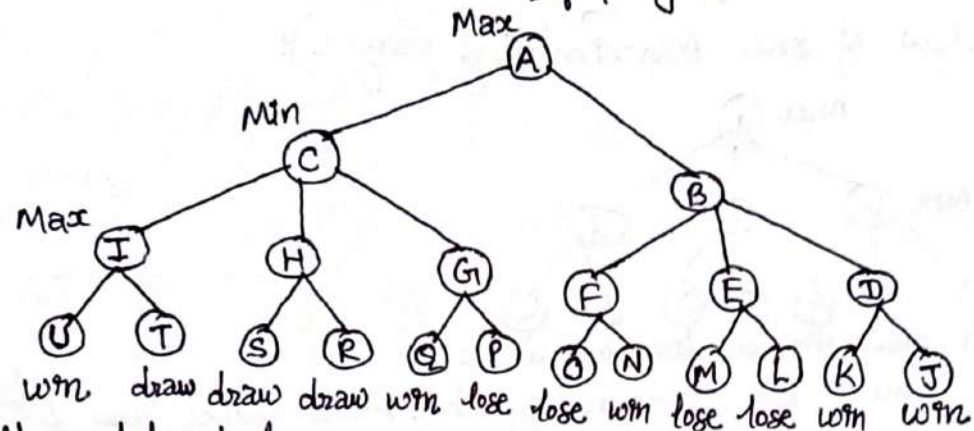
What move should be choosen by the Max player, and what should be the response of the Min player, assuming that both are using the mini-max procedure?

Solution:



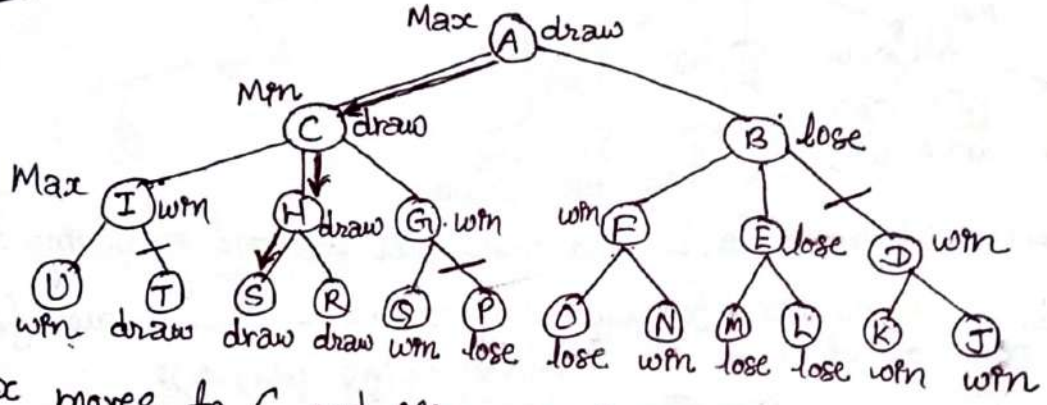
~~Max will move to~~ The move is shown by the arrow in above figure.

Example 3:- Consider the following two-person game tree, drawn from the point of view of the Maximizing player:



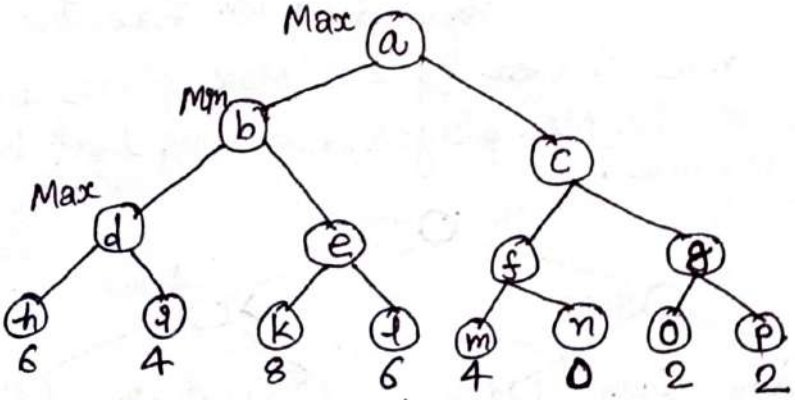
Use the alpha-beta ~~pruing~~ pruning procedure to determine the moves which should be choosen by two players, assuming a depth-first search of the tree. Which nodes need not to be examined?

Solution:



Max moves to C and Min moves to H and Max moves to S or R. The game ends in draw. The nodes ~~draw~~ T, P and D does not need to be examined.

Example 4:- Consider the following game tree (drawn from the point of view of the Maximizing player):

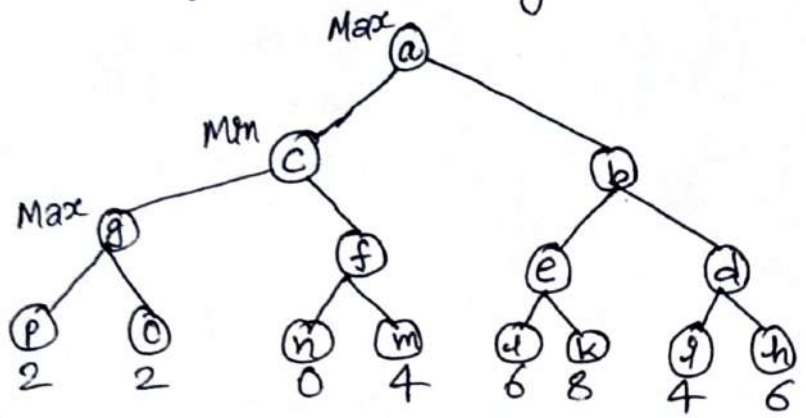


a) Use the mini-max procedure and show what moves should be choosen by the two players.

b) Use the alpha-beta pruning procedure and show what nodes would not need to be examined.

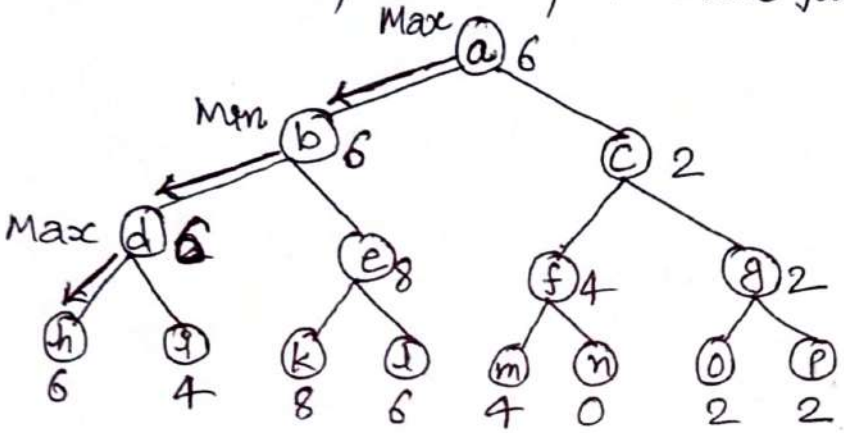
c) Consider the mirror image of above tree and apply again the alpha-beta pruning procedure. What do you notice?

The mirror image is the following tree:

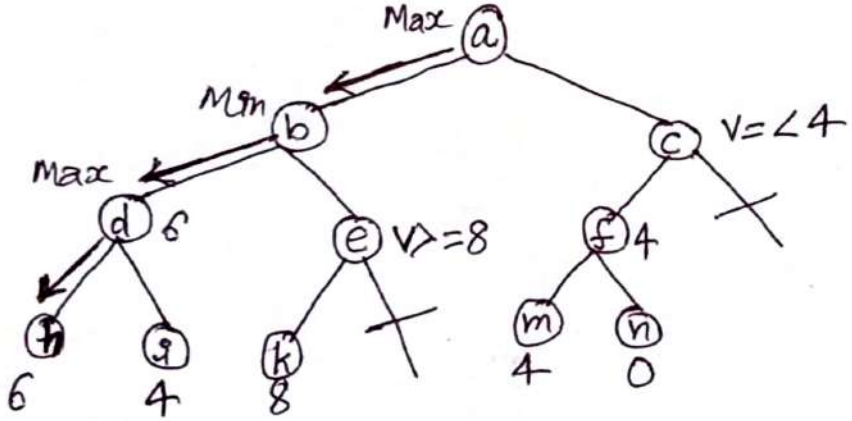


Solution:

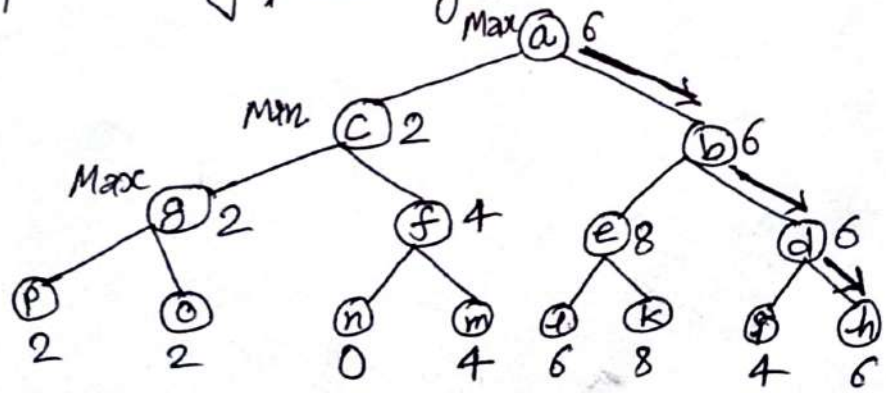
a) The mini-max procedure produce the following tree.



b) The cut branches need not be examined.



c) Using alpha-beta pruning on the mirror image of the tree does not produce any savings:



UNIT-4

Knowledge Representation

Knowledge → Knowledge is the information about a domain that can be used to solve problems in that domain. OR It is the sum of what is currently known. It consists of information that has been integrated, categorised, applied, experienced, revised etc. It is not just a data it also consists of facts, ideas, beleaves, association rules, relationships etc.

Importance of Knowledge: (In AI)

i) Knowledge is important in problem solving by computer.

ii) It is possible for an agents or systems to act accurately on some input only when it has the knowledge or experience about the input.

Issue in knowledge Representations:

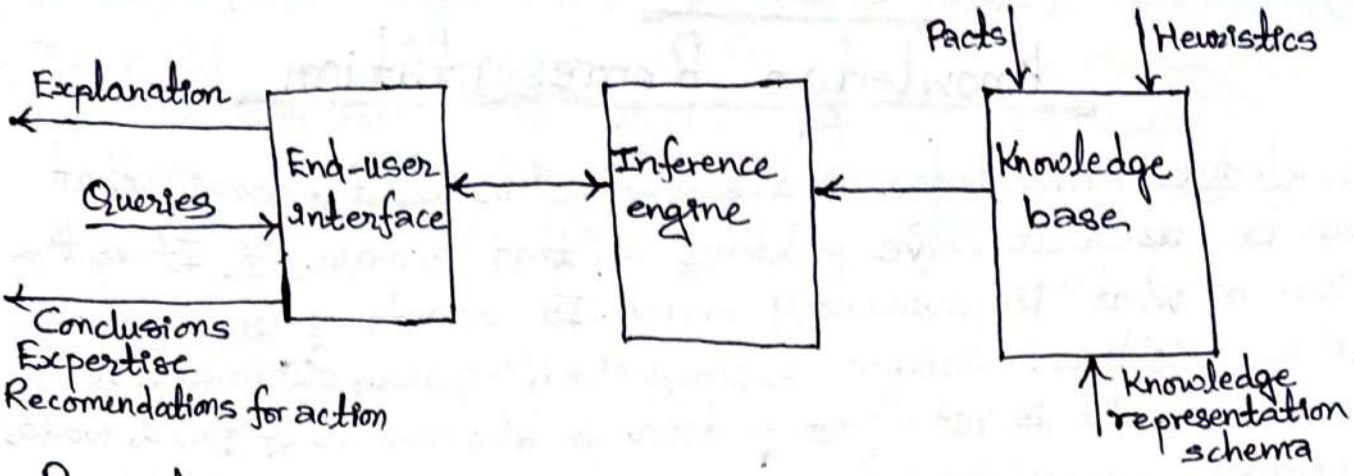
The process of converting informal knowledge into computer understandable form is known as knowledge representation. To solve any problem by a computer, first it must be represented in computer understandable form. Some of the issues regarding knowledge representation can be formulated as questions as follows:-

→ How can the problem be represented?
→ What specific knowledge about the world is required?
→ How can an agent acquire the knowledge from experts or from experience?
→ What distinctions in the world are needed to solve the problem?
→ How can the knowledge be debugged, maintained and improved?

Knowledge Representation System and its Properties:-

Knowledge Representation system is a formalized structure and sets of operations that contains the descriptions, relationships and procedures provided in an AI system. The process considers how to represent the knowledge in the system such that it can be used to solve problems. It basically consists of two components:

1) Knowledge base
2) Inference Methods.



Properties:-

i) Representatin adequacy:- Knowledge Representation system has the ability to represent all kinds of required knowledge.

ii) Inferential adequacy:- It is the ability to manipulate the knowledge represented to produce new knowledge corresponding from original knowledge.

iii) Inferential efficiency:- It is the ability to direct the inferential mechanisms into the most strong appropriate guides.

iv) Acquisitional efficiency:- It is the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

⊛. Types of Knowledge Representation Systems:-

1) Semantic Nets:- A sementic nets or network allows us to store knowledge in the form of a graphic (in the form of graphs) with nodes and arcs representing objects and their relationships. It could represent physical objects or concepts or even situations. This is simple and easy to implement and understand. It is more natural than logical representation. It allows us to categorize objects in various forms and link those objects.

Components of a Semantic Network:

i) Lexical part
- nodes → denoting objects
- links → denoting relation between objects
- labels → denote particular objects and relations.

ii) structural part: the links and nodes from directed graphs. The labels are placed on the links and nodes.

iii) Semantic part → Meanings are associated with the link and node lables.

iv) Procedural part →
constructors → allow creation of new links and nodes.
destructors → deletion of links and nodes
writers → allow creation and alteration of labels.
readers → can extract answers to questions.

Types of Semantic Network:

i) Definitional networks → These are designed to show relation between a concept type and a newly defined subtype. It supports the rule of inheritance for copying properties defined for a supertype to all of it's subtypes.

ii) Assertional networks → These are designed to assert positions. Unlike definitional networks, the information in an assertional network is assumed to be contingently true, unless it is explicitly marked with a modal operator.

iii) Implicational networks → These use implication as the primary relation for connecting nodes. They may be used to represent pattern of beliefs, causality or inferences.

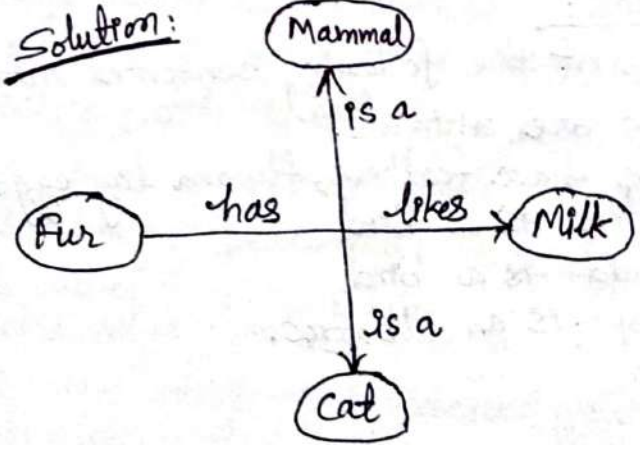
iv) Learning networks → These are used to build or extend their representations by acquiring knowledge from examples.

v) Hybrid networks → Combines two or more of the previous techniques, either in a single network or seperate, but closely interacting networks.

Examples of Semantic Network: (only for understanding)

i) General Example: Represent following statements in figure.
→ cat is a mammal.
→ cat likes milk
→ cat has fur
→ tommy is a cat,

Solution:



ii) AND/OR tree Example:-

AND/OR tree → It is an assignment of "True" or "False" to each of the leaves of tree.

→ An OR node is true if at least one of its children is true.

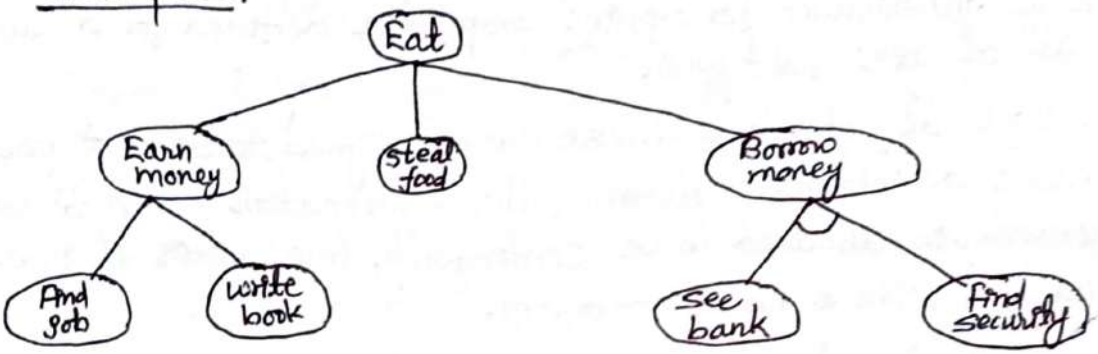
→ An AND node is true if all of its children are true.

Nodes:

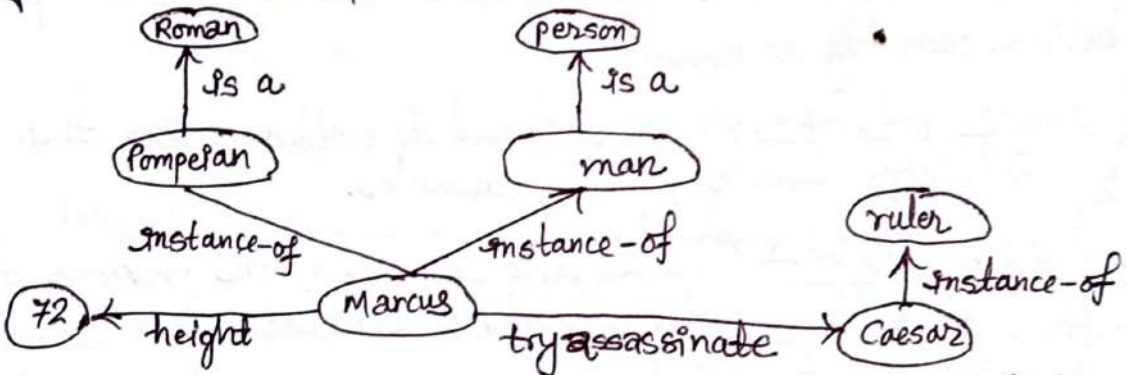
representation of AND node.

representation of OR node.

Example:



iii) Binary relations example:



(Marcus, Caesar) or constants (72) → represent individual objects

(Pompeian, Roman, man, person, ruler) → represent classes of individuals

instance-of → represents element of a class.

is a → represents subclass of a class.

try assassinate → represents tried to assassinate

⊛. Solved Examples:-

Example 1: Represent the following sentences into a semantic network.

→ Birds are animals.

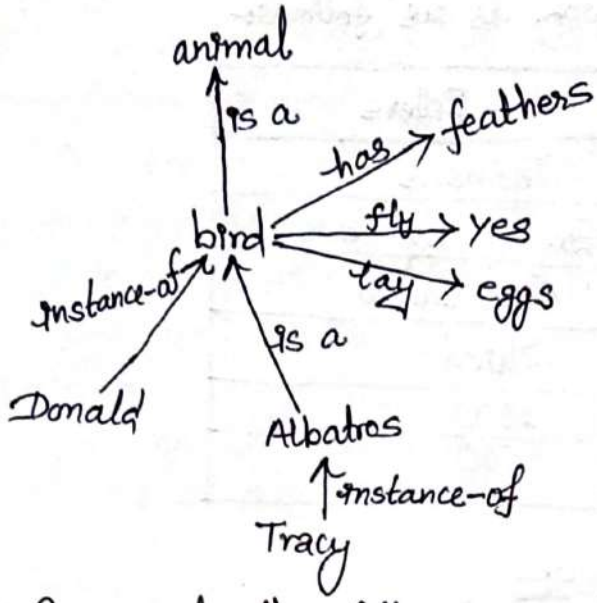
→ Birds have feathers, fly and lay eggs.

→ Albatros is a bird.

→ Donald is a bird.

→ Tracy is an albatros.

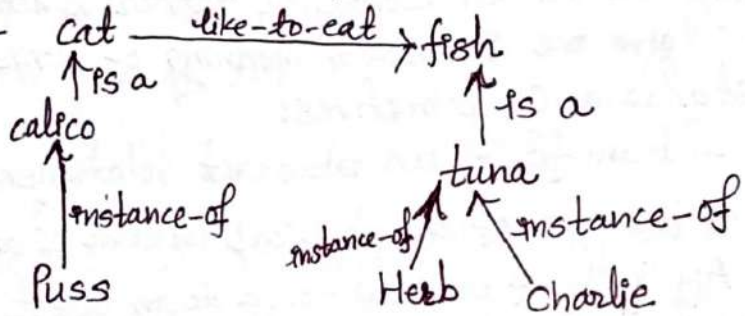
Solution:



Example 2: Represent the following sentences into a semantic network:

→ Puss is a calico.
→ Herb is a tuna
→ Charlie is a tuna
→ All tunas are fishes.
→ All calicos are cats.
→ All cats like to eat all kinds of fishes.

Solution:



2) Frame based Knowledge Representations:-

Frame is a type of schema used in many AI applications including vision and natural language processing. Frames provide a convenient structure for representing objects that are typical to a steratypical situations like visual scenes, structure of complex physical objects etc.

Frame systems usually have collection of frames connected to eachother. Each frame has a name and slots. Slots are the properties of the entity that has the name, and they have values or pointer to other frames. When the slots of a frame are all filled, the frame is said to be instantiated, it now represents a specific entity of the type defined by the unfilled frame. Empty frames are sometimes called object prototypes.

Slots having particular value may be:

→ a default value.
→ an inherited value from a higher-frame.
→ a specific value which might represent an exception.

Example:- A frame for a book is as follows:-

Slots	Fillers
publisher	Thomson
title	Expert Systems
author	Giarranto
edition	Third
year	1998
pages	600

3) Conceptual Dependencies:-

CD theory was developed to represent the meaning of Natural Language sentences. It helps in drawing inferences and and is independent of the language. CD representation of a sentence is not built using words in sentence rather built using conceptual primitives which give the intended meaning of words. Following are some of the standard CD primitives:

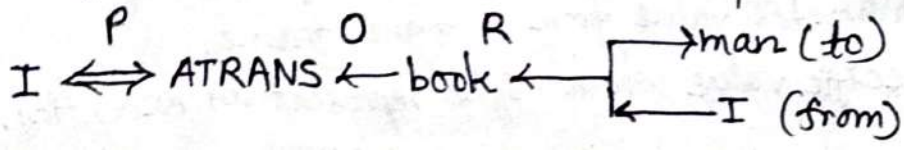
i) ATRANS → transfer of an abstract relationship (i.e, give).
ii) PTRANS → transfer of the physical location of an object (e.g, go).
iii) PROPEL → Application of physical force to an object (e.g, push).
iv) MOVE → Movement of a body part by it's owner (e.g, kick).
v) GRASP → Grasping of an object by an action (e.g. throw).
vi) INGEST → Ingesting of an object by an animal (e.g. eat).
vii) EXPEL → Expulsion of something from the body of an animal (e.g, cry)
viii) MTRANS → Transfer of mental information (e.g, tell).
ix) SPEAK → Producing of sounds (e.g. say).
x) ATTEND → Focusing of a sense organ toward a stimulus (e.g, listen).

There are four conceptual categories:

ACT → Actions {one of the CD primitives}
PP → Objects {picture producers}
AA → Modifiers of actions {action aiders}
PA → Modifiers of PP's {picture aiders}

Example 1:

I gave a book to the man. CD representation is as follows:



where,
→ Arrows indicate directions of dependency.
→ Double arrow indicates two way link between actor and action.

O → for the object case relation.
R → for the recipient case relation.
P → for past tense.
D → Destination.

Rule 1: PP ⇔ ACT
⇒ It describes the relationship between an actor and the event he or she causes.

Example 2: John ran.

CD representation is: John $\overset{P}{\Leftrightarrow}$ PTRANS

Rule 2: ACT ← PP
⇒ It describes the relationship between ACT and a PP(object) of ACT.

Example 3: John pushed the bike.

CD representation is: John ⇔ PROPEL ← bike

Rule 3: PP ↔ PP
⇒ It describes the relationship between two PP's, one of which belongs to the set defined by the other.

Example 4: John is doctor.

CD representation is: John ↔ doctor

Rule 4: PP ← PP
⇒ It describes the relationship between two PP's, one of which provides a particular kind of information about the other.

Example 5: John's dog.

CD representation is: dog $\overset{poss-by}{\leftarrow}$ John

Rule 5: PP ⇔ PA
⇒ It describes the relationship between a PP and a PA that is asserted to describe it.

Example 6: John is fat.

CD representation is: John ⇔ weight (>80).

Rule 6: PP ← PA.
⇒ It describes the relationship between a PP and an attribute that already has been predicated of it.

Problems with CD representation:-

i) It is difficult to construct original sentence from it's corresponding CD representation.
ii) Rules are to be carefully designed for each primitive action in order to obtain semantically correct interpretation.
iii) The CD representation becomes complex requiring lot of storage for many simple actions.
iv) It is difficult to find correct primitive in the given context.

4) Scripts (Script Structure):

The scripts are useful in describing certain sterotyped situations such as going to theater. It consists of set of slots containing default values along with some information about the type of values similar to frames. The values of the slots in scripts must be ordered and have more specialized roles.

Script Components:

i) Entry Conditions → Must be satisfied before events in the script can occur.
ii) Results → Conditions that will be true after events in script occur.
iii) Props → Slots representing objects involved in the events.
iv) Roles → Persons involved in the events.
v) Track → Specific variation on more general pattern in the script. Different tracks may share many components of the same script but not all.
vi) Scenes → The sequence of events that occur.

5) Rule Based Knowledge Representation System (RBS):-

A rule-based system (or production system) is a Knowledge Based System (KBS) in which the language is stored as rules; the rules come from human experts in a particular domain. It represents knowledge in the form of condition-action pairs called production rules.

→ If the condition C is satisfied then the action A is appropriate.

Examples:

→ If it is raining then open the umbrella.
→ If Cesar is a man then Cesar is a person.

In RBS the knowledge is seperated from AI reasoning processes, which means that new RBSs are easy to create. The syntax of rules is

IF <premise> THEN <action>

Components of RBS:

- Working Memory → Small memory into which only appropriate rules are copied.
- Rule base → To facilitate efficient access to the antecedent.
- Interpreter → Processing engine.

Example:- Production system for string sorting.

Problem: Sorting a string composed of letters a, b & c.

Short ~~Term~~ Term Memory: cbaca

Production Set:

1. ba → ab
2. ca → ac.
3. cb → bc.

Interpreter: Choose one rule according to some strategy.

Solution:

Iteration #	Memory	Conflict Set	Rule fixed.
0	cbaca	1,2,3	1
1	cabca	2	2
2	acbca	2,3	2
3	acbac	1,3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	∅	halt

6) Logic Based Knowledge Representations:-

@ Propositional Logic: (PL)

A proposition is a declarative statement which is either true or false. Propositional logic is the simplest form of logic where all the statements are made by propositions. It is a technique of knowledge representation in logical and mathematical form.

Example:

→ There are 7 days in a week (True Proposition)

→ 3+3 = 7 (False Proposition).

Note: Propositions are either true or false but not both.

Syntax of proportional logic:

The syntax of proportional logic is defined by the allowable sentence (atomic & composite). The sentence which is indivisible is called atomic sentence which consist of single proportion (that can be either true or false). Combination of two or more than two atomic sentences by using logical connectives (like AND, OR, NOT) is called complex or composite sentence.

Now proportional logic can be defined as; (read as 'not S')

If S is sentence → $\neg S$ is sentence (negative).

If S_1 & S_2 are sentences → $S_1 \wedge S_2$ is sentence (conjunction)

$S_1 \vee S_2$ is sentence (disjunction)

$S_1 \rightarrow S_2$ is sentence (implication)

$S_1 \leftrightarrow S_2$ is sentence (double implication).

Semantic of proportional logic:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Formal grammar for proportional logic:

Sentence → Atomic sentence / Complex sentence.

Atomic sentence → T / F / Symbol

Symbol → P / Q / R / ...

Complex sentence → $\neg$ Sentense / $S \wedge S$ / $S \vee S$ / $S \rightarrow S$ / $S \leftrightarrow S$

Logical Equivalence:

Two sentences α & β are logically equivalent ($\alpha \equiv \beta$) iff they are true in same set of models.

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination.

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition.

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination.

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$.

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$.

Validity in propositional logic:

A sentence is valid if it is true in all models. Valid sentences are also known as tautologies. Every valid sentence is logically equivalent to True.

e.g. $A \vee \neg A$

Q. Prove how $A \vee \neg A$ is valid? Use truth table.

Solution:

A	$\neg A$	$A \vee \neg A$
T	F	T } valid proved.
F	T	T

Satisfiability in propositional logic:

→ A sentence is satisfiable if it is true in some model.

e.g. $A \vee B$

→ A sentence is unsatisfiable if it is true in no model.

e.g. $A \wedge \neg A$.

Inference in propositional logic:

To infer the knowledge from KB mostly we use modus ponens.

i.e, $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$ & Elimination $\dfrac{\alpha \wedge \beta}{\wedge\alpha}$.

The set of entailed sentence can only increase ~~added~~ the information added to the KB, this process is called monotocity.

Inference using resolution:

1) Unit resolution rule:

$$\frac{l_1 \vee \ldots \vee l_{k1} \quad m}{l_1 \ldots \vee l_{j-1} \vee l_{j+1} \ldots \vee l_k}$$

where, l_j is literal.

2) Generalized resolution rule:

$$\frac{l_1 \vee \dots l_{k1}\ m_1 \vee_1 \dots \vee m_n}{l_a \vee \dots \vee l_{j-1} \vee l_{j+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

Resolution by using CNF:

A sentence that is expressed as conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF). Process for converting KB in CNF is as follows:

1) Eliminate biconditional (i.e, $\leftrightarrow$)
   e.g, $(B \leftrightarrow C) \equiv (B \rightarrow C) \wedge (C \rightarrow B)$
2) Eliminate implication (i.e, $\rightarrow$)
   e.g, $(B \rightarrow C) \equiv (\neg B \vee C)$
3) Move '$\neg$' inwards
   e.g, $\neg(A \wedge C) \equiv (\neg A \vee \neg C)$
4) Apply distributivity law ($\wedge$ over $\vee$) and flatten;
   e.g, $(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (A \vee C)$
   e.g, $((A \vee B) \vee C) \equiv (A \vee B \vee C)$

Q1. Convert Following KB into CNF.

i) $B \leftrightarrow (A \vee C)$

Solution:

$B \leftrightarrow (A \vee C)$
$\equiv (B \rightarrow (A \vee C) \wedge (A \vee C) \rightarrow B))$
$\equiv (\neg B \vee (A \vee C) \wedge (\neg (A \vee C) \vee B))$
$\equiv (\neg B \vee A \vee C) \wedge ((\neg A \vee B) \wedge (\neg C \vee B))$
$\equiv (\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

ii) $(A \rightarrow B) \vee (C \rightarrow B)$
$\equiv (\neg A \vee B) \vee (\neg C \vee B)$

iii) $(A \vee B) \rightarrow C$
$\equiv \neg (A \vee B) \vee C$
$\equiv (\neg A \wedge \neg B) \vee C$
$\equiv (\neg A \vee C) \wedge (\neg B \vee C)$

Resolution Algorithm:

Process (Algorithm)

i) Convert the given KB into CNF
ii) Add negation of the sentence to be entailed.
iii) Repeat the resolution rule.
iv) If there comes empty clause then.
   ↳ sentence is entailed to KB
   else
   ↳ sentence is not entailed.

Q1. Consider the knowledge base $KB = (B \Leftrightarrow (A \vee C) \wedge \neg B)$.
Prove that $\neg A$ can be inferred from above KB by using resolution.

Solution:

Step1: Converting KB into CNF

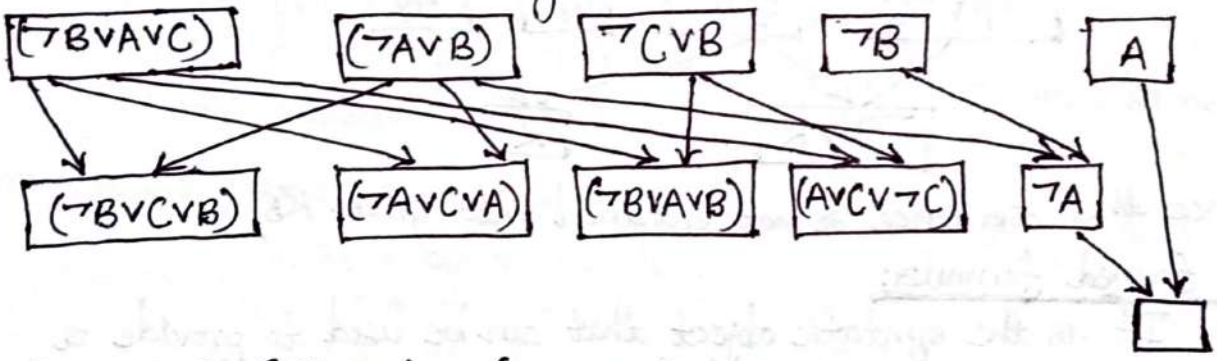
$$B \Leftrightarrow (A \vee C) \wedge \neg B$$
$$\equiv B \Rightarrow (A \vee C) \wedge (\neg(A \vee C) \Rightarrow B) \wedge \neg B$$
$$\equiv (\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B) \wedge \neg B$$
$$\equiv (\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B) \wedge \neg B$$
$$\equiv (\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B) \wedge \neg B$$

(Note that $\Leftrightarrow$ is same as $\leftrightarrow$ and $\Rightarrow$ is same as $\rightarrow$ we can use any notation.)

Step2: Add negation of sentence to be inferred from KB into KB.
Now KB contains following sentences all in CNF.

$$\equiv (\neg B \vee A \vee C)$$
$$\equiv (\neg A \vee B)$$
$$\equiv (\neg C \vee B)$$
$$\equiv \neg B$$
$$\equiv A$$

Step3: Now use Resolution algorithm:



Q2. Given $KB = \{(G \vee H) \rightarrow (\neg J \wedge \neg K), G\}$. Show that the given KB entail $\neg J$ or not?

Solution:

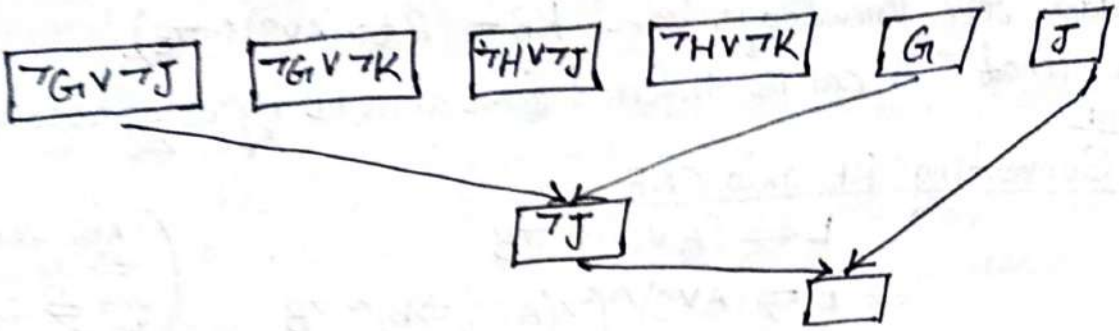
Step1:

$$(G \vee H) \rightarrow (\neg J \wedge \neg K)$$
$$\equiv (\neg(G \vee H) \vee (\neg J \wedge \neg K)$$
$$\equiv (\neg G \wedge \neg H) \vee (\neg J \wedge \neg K)$$
$$\equiv (\neg G \vee \neg J) \wedge (\neg G \vee \neg K) \wedge (\neg H \vee \neg J) \wedge (\neg H \vee \neg K)$$

Step2:

$$\neg(\neg J) \equiv J$$

Step3:



Hence the sentence is entailed into given KB.

Q3. KB = {P→¬Q, ¬Q→R}. Show that the sentence ¬P→Q is entailed from given CNF.

Solution:

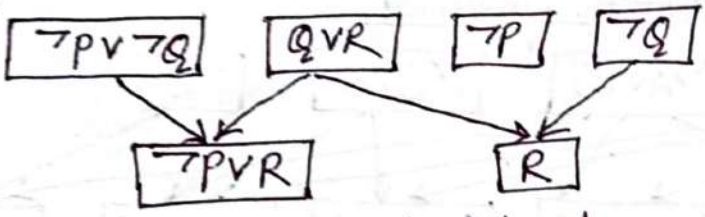
Step1:

P→¬Q	¬Q→R	¬P→Q
≡ ¬P∨¬Q	≡ Q∨R	≡ P∨Q

Step2:

P∨Q
≡ ¬P∧¬Q

Step3:



Hence the sentence is not entailed into given KB.

Well formed formula:

It is the syntatic object that can be used to provide a semantic meaning. A formal language can be considered to be identical to the inference iff they are well formed formula (wff). The wff are inductive defined as follows:

- ↳ each propositional variable is on its own a wff.
- ↳ if ϕ is a wff, then ¬ϕ is a wff.
- ↳ if ϕ & ψ are wff, & is binary connective, then (ϕ, ψ) is wff.

The wff for predicate logic is defined as:

- ↳ ¬ϕ is a wff when ϕ is a wff.
- ↳ (ϕ∧ψ) & (ϕ∨ψ) are wff when ϕ & ψ are wff.

⊛ Backward Chaining and Forward Chaining:

The completeness of resolution makes a very good inference model. But in many practical situation full power of resolution is not required. Real world KB often contains only horn clauses. A horn clause is disjunction of literal with at most one positive literal.

eg. $\neg A \vee B \vee \neg C$ (horn clause)
$A \vee B \vee C$ (not horn clause).

Properties of horn clause:
- → Can be written as implication.
- → Inference through forward and backward chaining.

A. Forward Chaining:

Forward chaining is the logical process of inferring unknown truths from known data and moving forward using determined conditions and rules to find a solution. It is also known as data driven reasoning.

Steps:
- → Given knowledge base of true facts.
- → Apply all rules that match facts in knowledge base.
- → Add conclusions to the knowledge base.
- → Repeat until goal is reached, OR repeat until no new facts added.

Example 1: Suppose we have three rules: (not imp)

R1: If A and B then D
R2: If B then C
R3: If C and D then E

⇒ If facts A and B are present, we infer D from R1 and infer C from R2. With D and C inferred, we now infer E from R3.

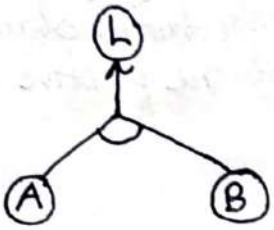
Q1. $P \rightarrow Q$
$L \wedge H \rightarrow P$
$B \wedge L \rightarrow H$
$A \wedge P \rightarrow L$
$A \wedge B \rightarrow L$
A
B

Show that Q can be inferred from given KB.

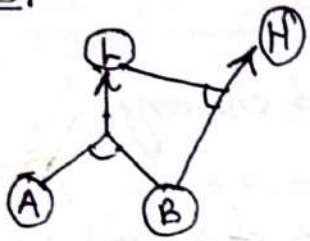
Solution:

Step1:

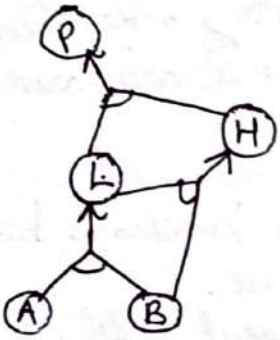
Step2:



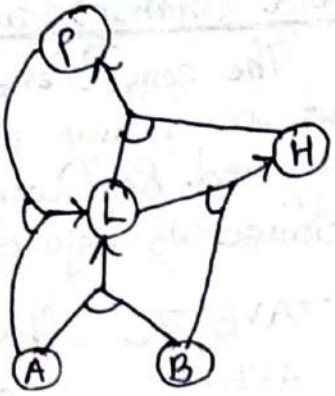
Step3:



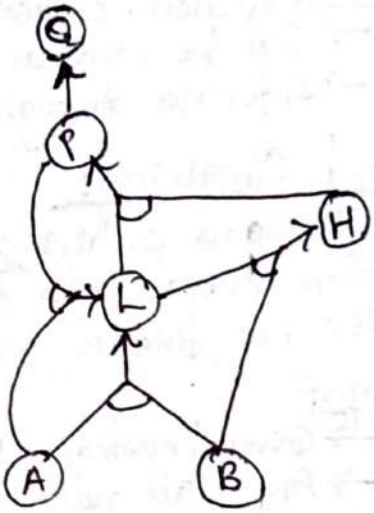
Step4:



Step5:



Step6:



B) Backward Chaining:-

Idea:

→ Work backward from Query Q to prove backward chaining.
→ Check if q is known already or prove by backward chaining.
→ Avoid loops: Check if new sub-goal is already on the goal state.

Q. $P \rightarrow Q$
$L \wedge H \rightarrow P$
$B \wedge L \rightarrow H$
$A \wedge P \rightarrow L$
$A \wedge B \rightarrow L$
A
B

Show that Q can be inferred from given KB by using backward chaining.

Solution:

We know that $P \rightarrow Q$, now we will try to prove P is true.
Again, $L \wedge H \rightarrow P$, try to prove L and H are true.

$B \wedge L \rightarrow H$
$A \wedge P \rightarrow L$ } try to prove B, L, A and P are true.

Now, A & B are already known since $A \wedge B \rightarrow L$, so L is known.
Since $B \wedge L \rightarrow H$, now H is also known.
Again, $L \wedge H \rightarrow P$, now P is also known/true.
Hence, Q can be inferred from given KB.

b) ⊛ Predicate Logic:- FOPL (First-Order Predicate Logic):

FOPL is the symbolized reasoning in which each sentence or statement is broken down into a subset of predicate, where predicate is considered as function & relationship of subject. It is also a bi-valued function i.e, $P: X \rightarrow \{\text{True or False}\}$ is called predicate on value x. A sentence in FOPL is written in the form of Px or P(x), where P is predicate and x is subject.

e.g;
man (Ram)
brother (Ram, Hari)
married (Puran, Sita)
brother of (Ram) = Hari

FOPL Syntax:

Sentence → Atomic sentence / Quantifier / ...sentence / ¬sentence
Atomic sentence → Predicate (term) / Term → Term
Term → function (term) / Term → Term
Connective → $\wedge$ / $\vee$ / $\neg$ / $\rightarrow$ / $\leftrightarrow$
Quantifier → $\forall$, $\exists$
Constant → A, B, C, X_1, X_2, Ram, Hari, ...
Variable → x_1, x_2, counter position, ...
Predicate → brother, has colour, ...
function → father of, sqrt, cosine, ...

FOPL Semantic:

An interpretation is required to give semantics FOPL. An interpretation is a non-empty set of objects. An interpretation provides:
→ Constant symbol to an object in the domain
→ Function symbols.
→ Predicate symbols.
then we can define universal & existential quantifiers.

Object/Subject: could be specified or unspecified.

Normally predicates start with capital letters.

eg:

Loves (Ram, Sita)
Loves (x, y).
Sunny (Friday)
hot (x) ↔ Hot (day)

Quantifiers (Quantification):

Quantifiers allow us to express properties of collection of objects instead of enumerating objects by name. There are two types of quantifiers. They are:

1) Universal quantifier (for all i.e ∀):

∀ <variable> <sentence>

e.g. Everyone at Texas is smart.

⇒ ∀ x AT(x, Texax) → Smart(x)

It is logically equivalent to the conjunction of instantiations of P.

i.e, AT (Arjun, Texas) → Smart(Arjun) ∧ AT (Karishma, Texas).

It is also equivalent to a set of implications over all objects.

Note: '→' is the main connective of ∀.

Common mistake of ∀

∀x AT (x, Texas) ∧ Smart (x)
(∧ ↑ mistake)

2) Existential Quantification (There exists i.e, ∃):

∃x <variable> <sentence>

e.g. Someone at Texas is smart.

⇒ ∃x AT (Texas) ∧ Smart (x).

It is logically equivalent to the disjunction of instantiations of P.

i.e, AT (Arjun, Texas) ∧ Smart (Arjun) ∨ AT (Karishma, Texas) ∧ Smart (Karishma).

Common mistake to ∃

∃x AT (x, Texas) → Smart (x)
(↑ mistake.)

Representing Knowledge in FOPL:

The objects from real world are represented by the constants (A, B, C., Ram, ...). For instance, the symbol "Tom" may represent a certain individual Tom.

→ Properties of object may be represented by predicates.
e.g, danger(Tom)

→ Predicate also represents relation between objects.
e.g, friends (Hari, Ram)

→ Predicate also represent in the form of Boolean constant.
e.g, sister (Arjun, Sristhi) → F

Properties of quantifiers:-

i) $\forall x \forall y \equiv \forall y \forall x$

ii) $\exists x \exists y \equiv \exists y \exists x$

iii) $\exists x \forall y \not\equiv \forall y \exists x$

e.g. $\exists x \forall y$ loves (x, y)
$\forall y \exists x$ loves (x, y)
} Not logically equivalent.

Inference in FOPL:

FOPL inference can be done by converting the KB into the predicate logic and using propositional logic inference. The major task in inference of FOPL is to deal with quantifiers so instantiation is required. There are two types of instantiation. They are:

1) Universal Instantiation (UI):

Substitute ground item (term without variable) for the variable.

e.g. $\forall x$ King$(x) \wedge$ Greedy$(x) \rightarrow$ Evil(x)
King (John), Greedy (John), Brother (Richard, John), Father (Arjun, John).

Now its UI is:

King (John) ∧ Greedy (John) → Evil (John)
King (Richard) ∧ Greedy (Richard) → Evil (Richard)
King (Arjun) ∧ Greedy (Arjun) → Evil (Arjun)
King (father of (John))
King (brother of (John)).

2) Existential Instantiation (EI):

For any sentence & variable V in that, introduce a constant that is not in the KB (called skolem constant) and substitute that constant on V.

eg, $\exists x$ Crown$(x) \wedge$ Onhead$(x, John)$

After EI:

Crown$(c1) \wedge$ Onhead$(c1, John)$

where c1 is skolem constant.

Resolution of FOPL:

→ Same as proposition logic (PL)
→ Allows extended syntax such as quantifiers.
→ Define causual form for FOPL i.e, CNF
→ Eliminate quantifiers from causual form.
→ Adopt resolution procedure to cope with variables. i.e, unification.

Rule for converting into CNF:

1) Eliminate bi-implication & implication similarly as propositional logic.

2) Move negation inwards by De-Morgan's law.

$\neg \forall x\ P \rightarrow \exists x\ \neg P$

& $\exists x\ P \rightarrow \forall x\ \neg P$

3) Eliminate double negation.

$\neg(\forall x\ \neg P) \rightarrow \exists x\ P$.

$\neg(\exists x\ P) \rightarrow \forall x\ P$

4) Remove bound variables if necessary.

OR operation

$\forall x\ P(x) \vee \exists Q(x) \rightarrow \forall x\ [P(x) \vee \exists y\ Q(y)]$

5) Use equivalence to move quantifier to the left.

e.g $\forall x\ P(x) \wedge Q \rightarrow \forall x\ (P(x) \wedge Q)$

↳ where x does not belong to Q.

6) Skolemize:

$\exists x\ P(x) \rightarrow P(c1)$

$\forall x\ \exists y\ P(x,y) \rightarrow \forall x\ P(x, c1)$

7) The formula now has only universal quantifier & drop universal quantifier.

8) Use distribution law to get CNF.

Q. Convert following KB into CNF.(FOPL)

$$\forall x[\forall y P(x,y) \rightarrow \neg \forall x (Q(x,y) \rightarrow R(x,y))]$$

$$\equiv \forall x [\neg(\forall y \neg P(x,y) \vee \neg \forall y (Q(x) \rightarrow \neg R(y))]$$

$$\equiv \forall x [\exists y P(x,y) \vee \neg \forall y (\neg Q(x) \vee \neg R(y))]$$

$$\equiv \forall x [\exists y P(x,y) \vee \exists y (Q(x) \wedge R(y)]$$

$$\equiv \forall x [\exists y P(x,y) \vee \exists z (Q(x) \wedge R(z))]$$

$$\equiv \forall x [\exists y \exists z (P(x,y) \vee (Q(x) \wedge R(z))]$$

$$\equiv \forall x [P(x,g(x)) \vee (Q(x) \wedge R(f(x))]$$

$$\equiv P(x,g(x)) \vee (Q(x) \wedge R(f(x)))$$

$$\equiv (P(x,g(x)) \vee Q(x)) \wedge (P(x,g(x)) \vee R(f(x)))$$

Q. Convert the following sentences in FOPL.

1) Both TIC & TCHIT are in Network.
⇒ In(TIC, Texas) ∧ In(TCHIT, Texas)

2) Both CSIT & BCA are TU affiliate.
⇒ Affiliation(CSIT, TU) ∧ Affiliation(BCA, TU).

3) Manoj wrote "the girl I love."
⇒ Writes(Manoj) → loves(girl).

4) Either Manoj or Harish wrote "the girl I like".
⇒ Writes(Manoj) ∨ Writes(Harish) → likes(girl).

5) Puskal owns a copy of AI.
⇒ Owns(Puskal, Copy) → AI(copy).

6) Every song that Baibhav sings on Sainik was written by Baibhav.
⇒ ∀x Song(Baibhav, x) ∧ Sainik(x) → Writes(Baibhav).

7) Ram is a man.
⇒ Man(Ram).

8) America brough Alaska from Russia
⇒ bought(who, what, from)
bought(America, Alaska, Russia)

9) Jym collects everything
⇒ ∀x Collects(Jym, x).

10) Somebody collects something. ⇒ ∃x,y collects(x,y).

11) No stinky shoes are allowed.
⇒ $\exists x$ shoes$(x) \wedge$ stinky$(x) \wedge \neg$allowed(x).

12) Good people always have good friends.
⇒ $\forall x$ Person$(x) \wedge$ Good$(x) \rightarrow \exists y$ friend(x,y).

13) You can fool some of the people all the time, and all of people some of the time but you cannot fool all the people all the time.

⇒ $[\exists x \forall t$ (person$(x) \wedge$ time$(t) \rightarrow$ canfool$(x,t))$] $\wedge$ [$\forall x \exists t$ (person$(x) \wedge$ time$(t) \rightarrow$ canfool(x,t)] $\wedge$ [$\forall x \forall t$ (person$(x) \wedge$ time$(t) \rightarrow \neg$canfool(x,t)]

Unification and Lifting:

Atomic sentences → The most basic sentences in FOPL formed from predicate symbol followed by a parenthesis with sequence of terms.

e.g. Chinky is a cat
⇒ Cat(Chinky).

Complex sentence → Combination of atomic sentences using connectives.

Before unification and lifting we must have the idea of general modus ponens (GMP) in FOPL. i.e, for atomic sentence p_i, p_i' & q, where there is substitution θ such that SUBSET(θ, p_i) = SUBSET(θ, p_i') for all i.

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q)}{\text{where } p_i'\theta = p_i\theta \text{ for all } i.}$$

i.e, SUBSET(θ, q)

∴ n atomic sentences + one implication applying SUBSET(θ, q) yields the conclusion.

e.g. p_1' = King(John) p_2' = Greedy(y)
p_1 = King(x) p_2 = Greedy(x)
$\theta = \{x/John, y/John\}$ q = evil(x)

SUBSET(θ, q) is Evil(John).

⇒ The process of converting knowledge base (KB) into logically equivalent FOBL is called lifting.

Unification: Process we use to find substitution that makes different logical expression looks identical.

i.e, $Unify(\alpha, \beta) = \theta$ if $\alpha\theta = \theta\beta$.

e.g, Who does Ramesh know?

$Unify(knows(Ramesh, x)) \rightarrow knows(Ramesh, Sita)$

$\theta = \{x/Sita\}$

Inference using Resolution:

i) Convert all sentences in CNF.
ii) Negate conclusion & convert into CNF.
iii) Add negated conclusion to premise clause.
iv) Repeat until contradiction or no progress is made.

Question:

i) If something is intelligent, it has common sense.
ii) Deep blues do not have common sense.

Prove that Deep blue is not intelligent.

Solution:

1) $\forall x$ Intelligent$(x) \rightarrow$ Hascommonsense(x)
2) $\neg$Has common sense(Deep blue)
3) $\neg$Intelligent (Deep blue).

Step1: Converting into CNF:

$\equiv \forall x [\neg Intelligent(x) \vee Hascommonsense(x)]$

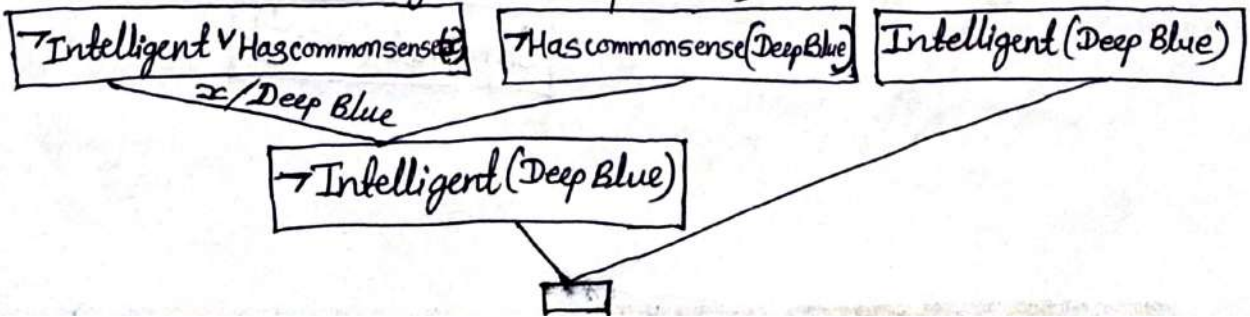
$\equiv \neg Intelligent(x) \vee Hascommonsense(x)$

Step2: Adding Negation to the conclusion.

$\neg(\neg Intelligent(Deep\ Blue))$

$\equiv Intelligent(Deep\ Blue)$

Step3:



Forward and Backward Chaining in FOPL:

Q. related to forward & backward chaining

Q. The law says that it is crime for an American to sell weapons to hostile nation. The country Nono, an enemy of America, has some missiles and all of its missiles were sold by Colonel West who is American. Now prove that Colonel West is criminal.

any small village of a country

Solution:

Representing in FOPL:

↳ It is a crime for American to sell weapons to hostile nation.

$$American(x) \wedge Weapon(y) \wedge sell(x,y,z) \wedge Hostile(z) \rightarrow Criminal(x).$$

↳ The country Nono, an enemy of America, has some missiles.

$$\exists x \; Owns(Nono,x) \wedge Missile(x).$$

FI: Replace x by M1.

$$Owns(Nono,M1) \wedge Missile(M1).$$

↳ All missiles are sold by Colonel West.

$$Missile(x) \wedge Owns(Nono,x) \rightarrow sells(west,x,Nono).$$

Now, we must know that missiles are weapons.

$$Missile(x) \rightarrow Weapon(x).$$

Again, enemy of America counts a hostile.

$$enemy(x,America) \rightarrow Hostile(x).$$

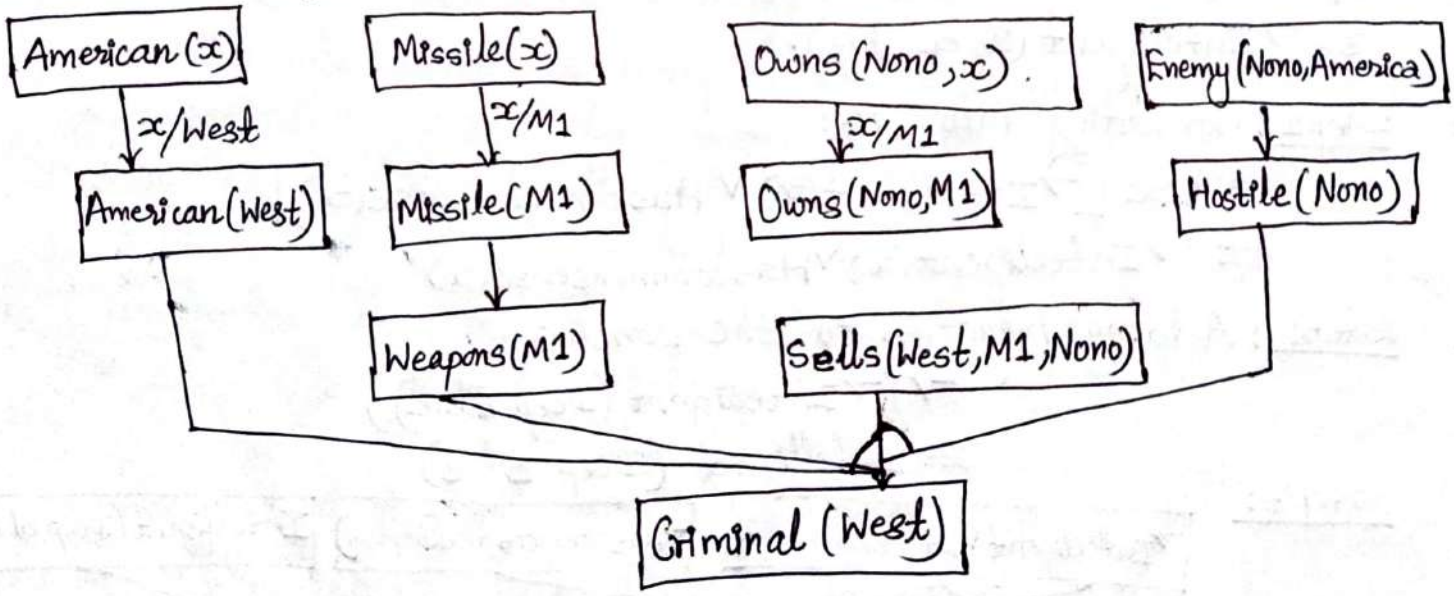
Also, Colonel West is an American.

$$American(West)$$

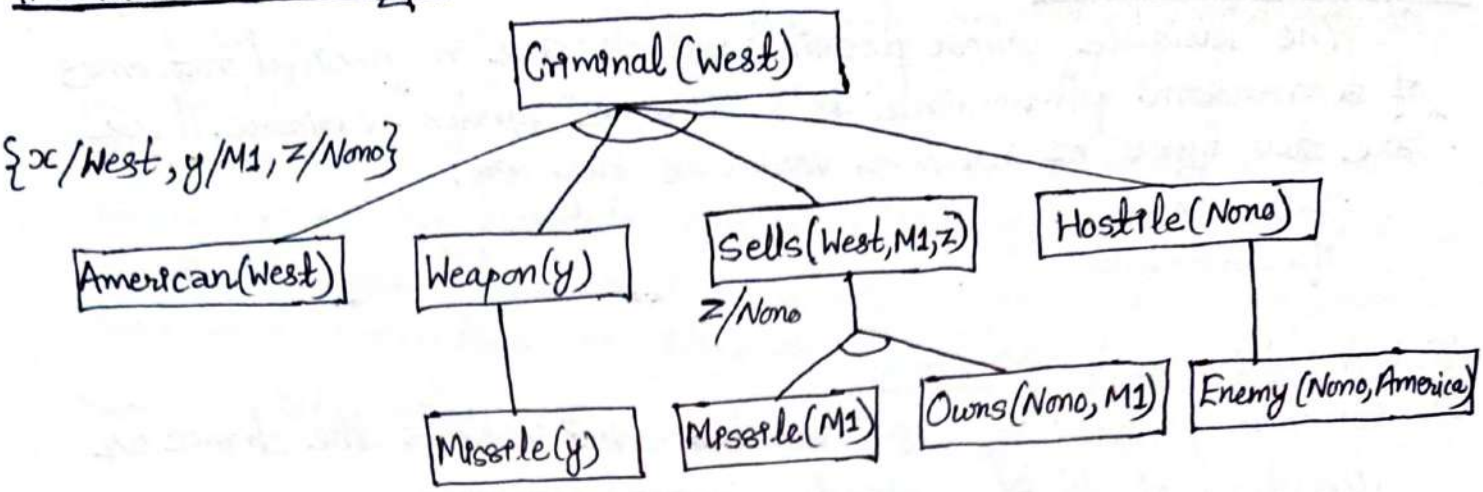
Again, the country Nono is a enemy of America.

$$enemy(Nono,America).$$

#Forward Chaining:



#Backward Chaining:



⊛. Handling Uncertain Knowledge:-

→ Agents need to handle uncertainity whether due to partial observability, non-determinism or combination of both. An agent may never know for certain what state it in or where it will end up after sequence of actions.

→ Although agent already have the knowledge about environment but due to some external natural factor agent will not be able to achieve actual mathematical interpretation goal, so it must take decision by analysing external factors and that decision is called rational decision.

e.g. Automated taxi

→ Let us take another example of uncertain knowledge in simple rule.

$$\text{Toothache} \rightarrow \text{Cavity}$$

The rule itself is wrong in real scenario. Not all patients with toothache have cavities, some of them have gum disease or any other problems. Now the rule becomes:

$$\text{Toothache} \rightarrow \text{Cavity} \vee \text{Gum problem} \vee \ldots$$

→ Unfortunately in order to make rule true we have to add almost unlimited list of possible problems.

→ So to move from one state to another state any fixed rule is applied, this case is considered as condition of uncertainity.

→ The only decision of uncertainity will be generated from probability theory.

⊛ Random Variables:

The variable whose possible values are numerical outcomes of a random phenomena is known as random variable. There are two types of random variables they are:

1) Discrete.
2) Continious.

⊛ Probability and it's types:

→ Simply probability is a number that reflects the chance or likelihood of that particular event occurance.

→ ~~It refers to~~ The degree of belief in proposition in the absence of any other information is called unconditional or prior probability.

Example:

$$P(A) = \frac{\text{total number of way that event A occur}}{\text{total number of possible outcomes.}}$$

$$P(\text{cavity}) = \frac{\text{cavity}}{\text{Toothache}}$$

→ The probability of the event occuring with some relationship to one or more other events is called conditional or posterior probability.

Example:

(representing age interval like young, adult, teenage)

$$P(\text{cavity} / \text{Teen}) = \frac{\text{Cavity} \wedge \text{Teen}}{\text{Toothache}}$$

→ Decision theory = Probability theory + Utility theory.

⊛ Inference using Full Joint Distribution:

The process of generating knowledge by using joint probability is called an inference using full joint probability distribution. It is the statistical measure that calculate the likelihood of two events occuring together & at the same point of time.

Example:- Let we have the following set of domains:

Age = {Child, Teen, Young, Aduld} & Cavity {T, F}

Now, the probability of P(Age, Cavity) is represented in 4X2 matrix as:

Age	Child	Teen	Young	Adult
Cavity=T	0.144	0.02	0.016	0.02
Cavity=F	0.576	0.08	0.064	0.08

⊛ Baye's Rule and its uses:

Baye's theorem is a way to apply conditional probability for prediction. Conditional probability is the probability of an event happening, given that it has some relationship to one or more other events. For example, the probability of getting a parking space is connected to the time of day we park, where we park and what conventions are going on at any time.

Mathematically, Bayes theorem is defined as:

$$P(A/B) = \frac{(B/A)\,P(A)}{P(B)}$$

Uses:
→ It is widely used to provide probabilistic prediction in AI.
→ In finance, Baye's theorem can be used to rate the risk of lending money to potential borrowers.
→ It allows us to update predicted probabilities of an event by incorporating new information.

Example: A doctor knows that the disease meningitis causes the patient to have a stiff neck 50% of the time. The doctor also knows that the probability that a patient has meningitis is 1/50,000 and the probability that any patient has a stiff neck is 1/20. Find the probability that a patient with a stiff neck has meningitis.

Solution: Let m be the patient having Meningistis and s be the patient having stiff neck.

Given,

$$P(s/m) = 50\% = 0.5$$
$$P(m) = 1/50{,}000$$
$$P(s) = 1/20$$

Now,

$$P(m/s) = \frac{P(s/m)\cdot P(m)}{P(s)} = \frac{0.5 \times \frac{1}{50{,}000}}{1/20} = 0.0002.$$

⊛. Bayesian Networks:

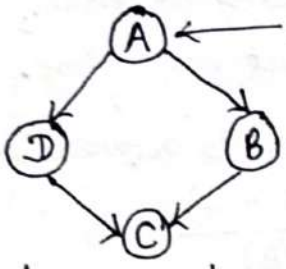
Bayesian network is a type of probabilistic graphical model that uses Bayesian inference for probabilistic inference. It is a directed cyclic graph which consist of:

* a set of random variables represented as nodes of network.

ii) Set of directed links connecting pair of nodes. If there is an arrow from node x to y then x is called parent of y.

iii) Each x_i has a conditional probability distribution ($P(x_i)$/Parents(x_i)) that quantifies the effect of the parents on that node.

Example:

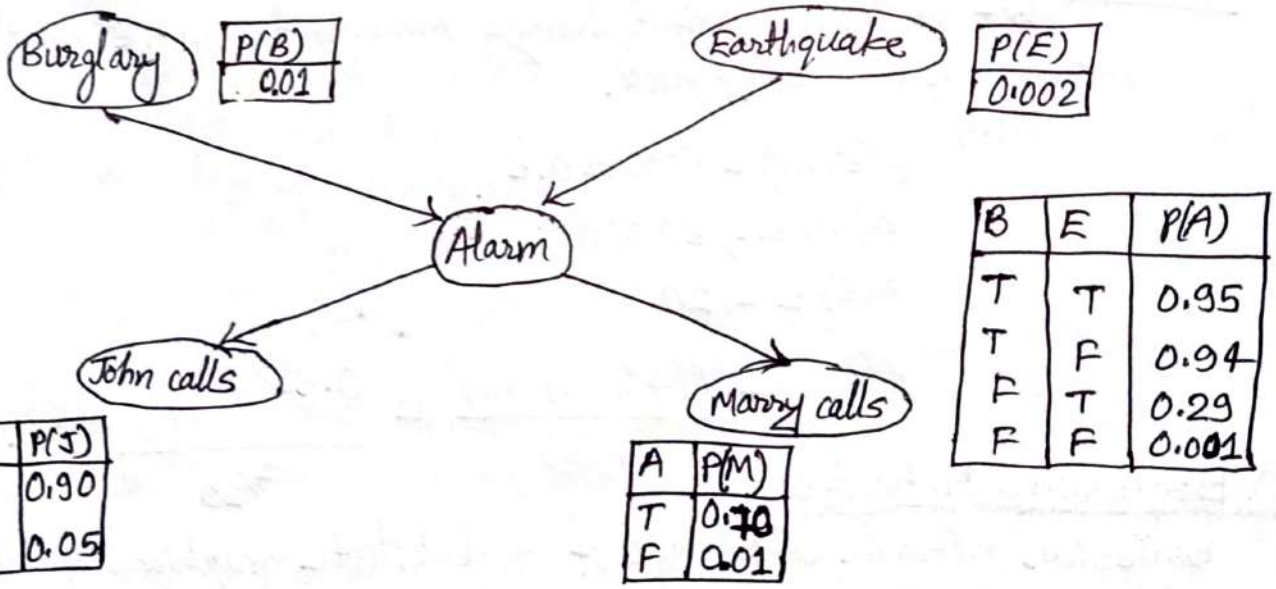


← A is the parent of D & B

← The conditional probability distribution is calculated to find the effect of A in B.

⇒ Bays network is used to find the relationship of dependent variable on uncertain condition.

Example: Let we have a new burglar alarm installed at home. It is reliable at detecting a burglary but also responds on the occasion of minor earthquakes. We also have two neighbours Johny and Mary who have promised us to call when we are at our work if they hear the alarm. John nearly calls when he hears the alarm but sometime confuses the telephone ringing with the alarm & call them too. Mary on the other hand likes rather loud music & often misses the alarm. Altogether given the evidence of who has or has not called the probability on Bayesian network is:



Here, the probability of alarm depends on the burglary & earthquake but the calls of John & Mary only depends on the alarm.

⊛. Inference with Bayesian (Belief) Network:-

First we simply evaluate the joint probability of a particular assignment of values for each variable (or a subset) in the network. For this, we already have a factorized form of the joint distribution, so we simply evaluate the product using the provided conditional probabilities.

Example: What is the probability that the alarm has sounded, but neither burglary nor an earthquake has occured, and both John and Marry call?

(continue Q. of previous eg) (from prerious example)

Solution:-

Given,

$P(B) = 0.01$

$P(\neg B) = 1 - 0.01 = 0.99$

$P(E) = 0.002$

$P(\neg E) = 1 - 0.002 = 0.998$

$P(A/\neg B, \neg E) = 0.001$

$P(J/A) = 0.90$

$P(M/A) = 0.70$

Therefore the probability of above condition is given by;

$P(\neg B) \times P(\neg E) \times P(A/\neg B, \neg E) \times P(J/A) \times P(M/A)$

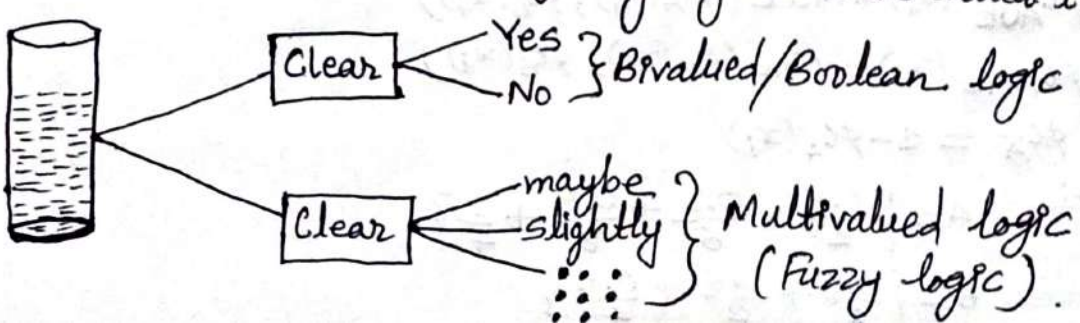
$= 0.99 \times 0.998 \times 0.01 \times 0.9 \times 0.7$

$= 0.00062$

⊛ Fuzzy Logic :-

The word fuzzy refers to things which are not clear. Any event, process, or function that is changing continiously cannot always be defined as either true or false, which means that we need to define such activities in a Fuzzy manner.

In fuzzy systems, the values are indicated by a number in the range from 0 to 1. Here 1.0 represents absolute truth and 0.0 represents absolute fa" eness. The number which indicates the value in fuzzy systems is called truth value.



⊛. Fuzzy sets:

(i.e, classical set)

A set is an unordered collection of different elements. If the order of the elements is changed or any element of a set is repeated, it does not make any changes in the set.

Fuzzy sets can be considered as an extension and gross over simplification of classical sets. Classical set contains elements that satisfy precise properties of membership while fuzzy set contains elements that satisfy imprecise properties of membership. A set X in which each element y has a grade of membership $\mu_x(y)$ in the range 0 to 1, i.e, set membership may be partial.

For example:- If cold is a fuzzy set, exact temperature values might be mapped to the fuzzy set as follows:

15 degrees → 0.2 (slightly cold)

10 degrees → 0.5 (quite cold)

0 degrees → 1 (totally cold).

Membership:

Membership of each element in fuzzy set is mapped as $\mu(x) = [0,1]$.

If A is a fuzzy set over domain $X = \{a, b, c\}$ then A is defined as;

$$A = \left\{ \frac{\mu_A(a)}{a} + \frac{\mu_A(b)}{b} + \frac{\mu_A(c)}{c} \right\}.$$

e.g $\text{Even} = \left\{ \frac{0.9}{1} + \frac{1}{2} + \frac{0.98}{3} \right\}$ or

$$= \left\{ \frac{0.9}{1}, \frac{1}{2}, \frac{0.98}{3} \right\} \text{ or}$$

$$= \{(1, 0.9), (2, 1), (3, 0.98)\}$$

Fuzzy Set Operations:-

i) Union

ii) Intersection

iii) Complement

Suppose A & B over X then;

$$\mu_{A \cup B}(x_i) = \text{Max}(\mu_A(x_i), \mu_B(x_i))$$

$$\mu_{A \cap B}(x_i) = \text{Min}(\mu_A(x_i), \mu_B(x_i))$$

$$\mu_{\bar{A}} = 1 - \mu_A(x_i).$$

Example: $A = \left\{ \frac{0.5}{1} + \frac{0.2}{6} + \frac{0.9}{8} + \frac{1}{4} \right\}$

$$B = \left\{ \frac{0.9}{1} + \frac{0.6}{2} + \frac{0.5}{4} \right\}$$

$$A \cup B = \left\{ \frac{MAX(0.5,0.9)}{1} + \frac{MAX(0,0.6)}{2} + \frac{MAX(1,0.5)}{4} + \frac{MAX(0.2,0)}{6} + \frac{MAX(0.9,0)}{8} \right\}$$

$$= \left\{ \frac{0.9}{1} + \frac{0.6}{2} + \frac{1}{4} + \frac{0.2}{6} + \frac{0.9}{8} \right\}$$

$$A \cap B = \left\{ \frac{MIN(0.5,0.9)}{1} + \frac{MIN(0,0.6)}{2} + \frac{MIN(1,0.5)}{4} + \frac{MIN(0.2,0)}{6} + \frac{MIN(0.9,0)}{8} \right\}$$

$$= \left\{ \frac{0.5}{1} + \frac{0}{2} + \frac{0.5}{4} + \frac{0}{6} + \frac{0}{8} \right\}$$

$$\mu_{\bar{A}} = \left\{ \frac{1-0.5}{1} + \frac{1-0.2}{6} + \frac{1-0.9}{8} + \frac{1-1}{4} \right\}$$

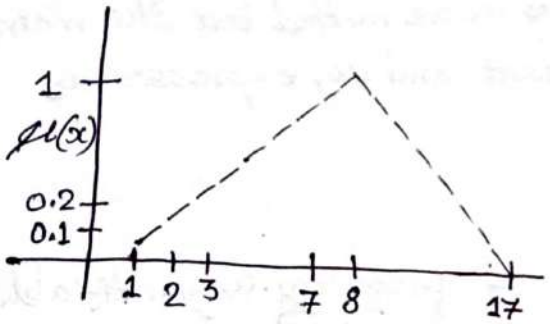
$$= \left\{ \frac{0.5}{1} + \frac{0.8}{6} + \frac{0.1}{8} + \frac{0}{4} \right\}$$

Q. Verify $\overline{A \cup B} = \bar{A} \cap \bar{B}$ for above example.

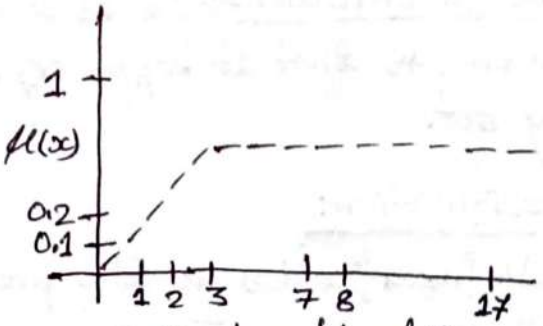
each value को लागि for eg. $\frac{1-0.9}{1}$ for first + ...

Here,
→ माथिको जस्तै $\bar{B} = \mu_{\bar{B}}$ calculate गर्ने.
→ त्यसपछि माथिको $A \cup B$ बाट $(1 - A \cup B)$ गरि $\overline{A \cup B}$ निकाल्ने (यो LHS भयो Q. को).
→ Finally RHS calculate गर्ने $\bar{A} \cap \bar{B}$, $\bar{A}$ र $\bar{B}$ first calculate गरि (यो RHS).
→ LHS र RHS equal आउछ।.

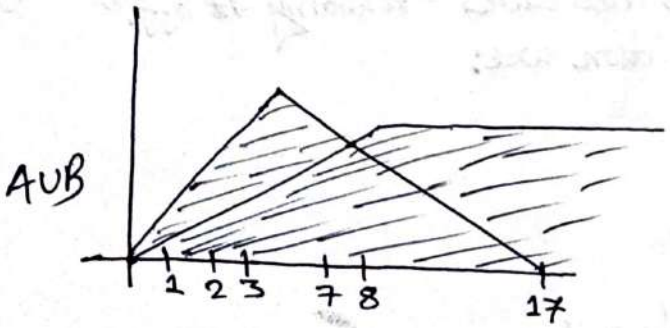
⊛ Membership graph of Fuzzy set.

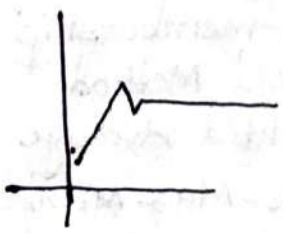


Membership of A

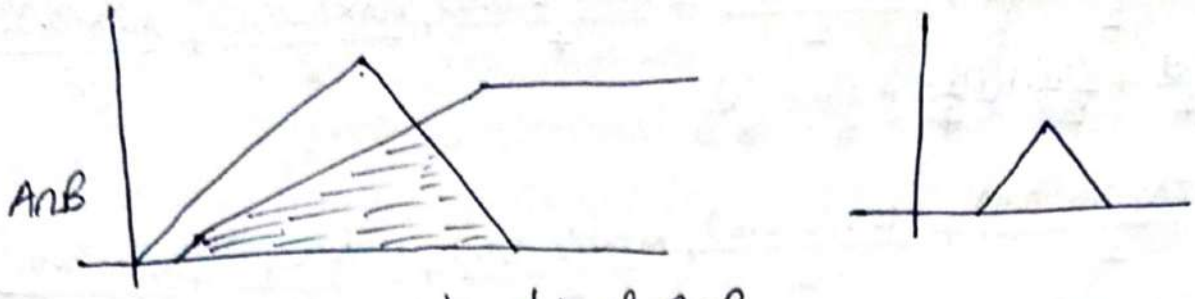


Membership of B





Membership of A∪B



Membership of A∩B

⊛. Fuzzification:

Fuzzification is the process ~~in which~~ where membership functions are applied, and the degree of membership is determined. There are two important methods of fuzzification.

i) Support fuzzification → In this method, the fuzzified set can be expressed with the help of following relation:

$$A^{\sim} = \mu 1\, Q(x1) + \mu 2\, Q(x2) + \ldots + \mu n\, Q(xn).$$

where, $Q(x_i)$ → kernel of fuzzification.

This method is implemented by keeping μ_i constant and x_i being transformed to a fuzzy set $Q(x_i)$.

ii) Grade fuzzification → It is similar to above method but the main difference is that it keeps x_i constant and μ_i expressed as fuzzy set.

⊛. Defuzzification:

Defuzzification is the process of producing a quantifiable result in fuzzy logic. It interprets the membership degrees in the fuzzy sets into a specific action or real-value. Mathematically the process of Defuzzification is also called "rounding it off."

The different methods of Defuzzification are:

i) Max-membership method
ii) Centroid Method
iii) Weighted Average Method
iv) Mean-Max Membership.

⊛. Applications of fuzzy logic systems:

→ Used in aerospace field for altitude control of spacecraft and satellite.
→ Used in automotive system for speed control, traffic control.
→ Used in Natural language processing.
→ Used in modern control systems such as expert systems.
→ Used in Neural Networks.

⊛ Advantages of Fuzzy logic system:

→ This system can work with any type of inputs whether it is imprecise, distorted or noisy input information.
→ The construction of Fuzzy logic Systems is easy and understandable.
→ The algorithms can be described with little data, so little memory is required.
→ Fuzzy logic comes with mathematical concepts of set theory and the reasoning of that is quite simple.

⊛. Disadvantages of Fuzzy logic systems:

→ It leads to ambiguity and there is no systematic approach to solve a given problem.
→ Proof of its characteristics is difficult or impossible in most cases.
→ Accuracy is compromised, since it works on precise as well as imprecise data.

Unit-5

Machine Learning:

Machine learning is a sub field of Artificial Intelligence that provides computer the ability to learn and improve its learning from experience without being written rules explicitly. The machine start learning with observations or data, in order to look for patterns in data and make better decisions in the future based on the examples that it has been provided.

Applications:

- → Computer Vision Processing
- → Natural Language Processing
- → Forecasting (e.g, stock market trends)
- → Games
- → Expert Systems.

The difference between traditional programming and machine learning can be illustrated with following figures:

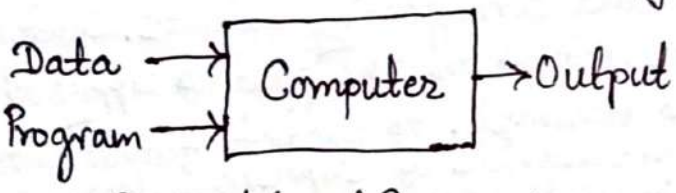


fig: Traditional Programming.

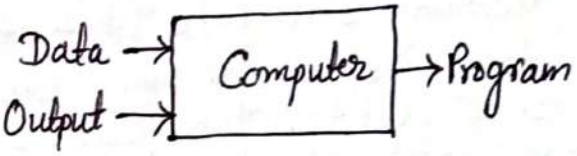
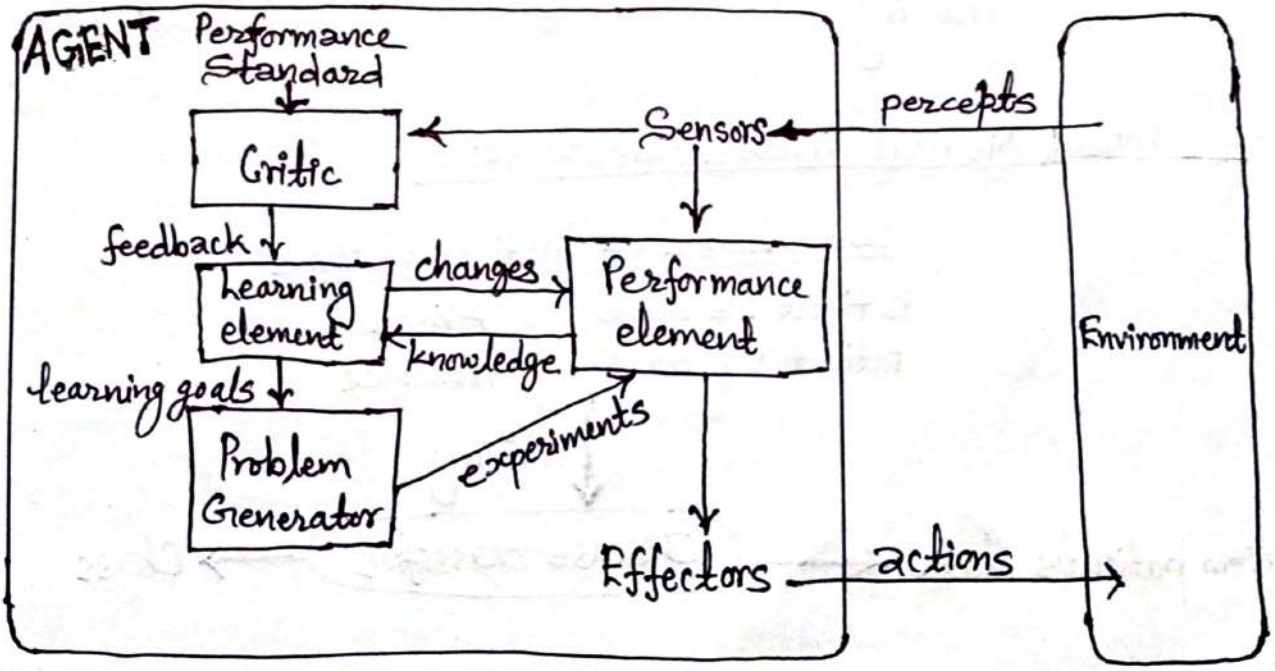


fig: Machine learning

⊛. Machine learning architecture or framework (Concept of Learning):



Learning agent consists of following components:

i) Learning element → It receives and processes the input obtained from environment.

ii) Knowledge base → It is a collection of knowledge. It updates knowledge periodically if agent is able to acquire new knowledge.

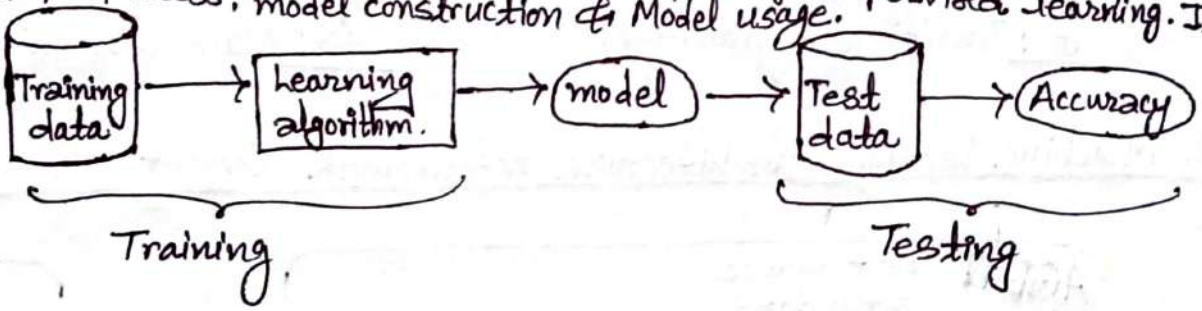
iii) Performance element → It uses the updated knowledge base to perform some tasks or solves some problems and produces the corresponding output.

iv) Feedback element → It receives two inputs, one from learning element and one from standard system. It is used to determine what should be done to produce the correct output.

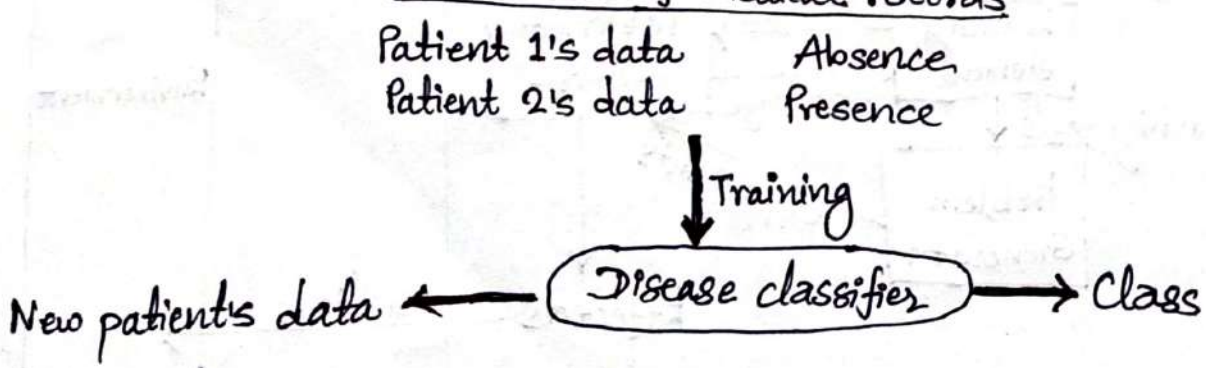
v) Standard system → It is a well programmed system that is able to produce the correct output.

⊛. Types of Machine Learning:

i) Supervised Learning → The system is supplied with a set of training examples consisting of inputs and corresponding outputs. Supervised learning can take what it has learnt in the past & apply that to a new data using labelled example to predict future patterns and events. Classification is an example of supervised learning. It is two-step process: model construction & Model usage.



Example: Medical disease classification.



ii) Unsupervised learning → It is also called predictive learning. In this learning information is used to train is neither classified nor labelled. It studies how system can infer a function to describe a hidden structure unlabelled data. This is similar to how babies learn easily in their life.

Examples: Clustering of data into similar group, Predict new target market etc.

iii) Semi-supervised learning → It falls somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training — typically a small amount of labelled data and a large amount of unlabelled data. This method are able to improve learning accuracy.

Example:- Classification of target market after searching information on GOOGLE.

iv) Reinforcement learning → It is a type of dynamic learning that trains agent using reward & punishment. In this learning, agent learns by interacting with environment & consists of three components: Agent (learner), Environment (Agent interacts with) and Action (what agent can do).

Example: FB Newsfeed, Youtube recomendations etc.

Types of Reinforcement: (less imp)

i) Positive → It increases the strength and the frequency of the behaviour. It has a positive effect on the behaviour.

Advantages: → Maximizes performance
→ Sustain change for long period of time.

Disadvantage: Overhead of states can diminish the results.

ii) Negative → Strengthening of a behaviour because a negative condition is stopped or avoided.

Advantage: → Increases behaviour

Disadvantage: Provides enough to meet up the minimum behaviour.

⊛. Statical-based Learning: Navie Bayes Model:-

Basics: ← for understanding and solving problems in easier way

i) Probability of event X i.e, $P(X) = \frac{\text{No. of times X occurs}}{\text{Total no. of events in sample space}}$

ii) **Joint Probability:** Consider two events A & B that occurs:

$P(A \wedge B) = P(A) * P(B)$ if the events are independent.

$P(A \wedge B) = P(A/B) * P(B)$

iii) **Bayes Theorem:** $P(A/B) = \frac{P(B/A)\,(PA)}{P(B)}$

← write from here if asked in exam

Navie Baye's Learning / Classifier:

In this model class variable C (which is to be predicted) is the root & the attribute variable x_i are the leaves. The model is called navie because that attributes are conditionally independent of each other in the given class. To calculate Navie Bayes learning we use Baye's theorem, which helps to calculate conditional dependent probability.

$$P(C/x_1 \ldots x_n) = \alpha\ P(x) \prod_{i=1}^{n} P(x_i/C)$$

This model is used to calculate the mostly likelihood parameters or attributes of a given class.

Applications of Navie Baye's Model:

→ Real time prediction.
→ Text classification.
→ Recommendation system.

Example:- From the given data set decide whether to play golf or not in today's conditions:

today = (Sunny, Hot, Normal, False).

S.N.	Outlook	Tempreature	Humidity	Wind	Play Golf
1.	Rainy	Hot	High	False	No
2.	Rainy	Hot	High	True	No
3.	Overcast	Hot	High	False	Yes
4.	Sunny	Mild	High	False	Yes
5.	Sunny	Cool	Normal	False	Yes
6.	Sunny	Cool	Normal	True	No
7.	Overcast	Cool	Normal	True	Yes
8.	Rainy	Mild	High	False	No.
9.	Rainy	Cool	Normal	False	Yes.
10.	Sunny	Mild	Normal	False	Yes
11.	Rainy	Mild	Normal	True	Yes
12.	Overcast	Mild	High	True	Yes.
13.	Overcast	Hot	Normal	False	Yes.
14.	Sunny	Mild	High	False	No.

Solution:

For Outlook:

	Yes	No	P(Yes)	P(No)
Sunny	3	2	3/9	2/5
Overcast	4	0	4/9	0
Rainy	2	3	2/9	3/5
Total	9	5		

For tempreature:

	Yes	No	P(Yes)	P(No)
Hot	2	2	2/9	2/5
Cool	3	1	3/9	1/5
Mild	4	2	4/9	2/5
Total	9	5		

For Humidity:

	Yes	No	P(Yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5		

For Wind:

	Yes	No	P(Yes)	P(No)
True	3	2	3/9	2/5
False	6	3	6/9	3/5
Total	9	5		

Since today's conditions are Sunny, Hot, Normal, False only according to the question

Now,

$$P(Yes/Today) = P(Sunny/Yes) \times P(Hot/Yes) \times P(Normal/Yes) \times P(False/Yes) \times P(Yes)$$

$$= \frac{3}{9} \times \frac{2}{9} \times \frac{6}{9} \times \frac{6}{9} \times \frac{9}{14}$$

Total no. of Yes / Total Yes + Total No

$$= 0.0212$$

$$P(No/Today) = P(Sunny/No) \times P(Hot/No) \times P(Normal/No) \times P(False/No) \times P(No)$$

$$= \frac{2}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{3}{5} \times \frac{5}{14}$$

$$= 0.0069.$$

Since, sum of probabilities must be equal to 1. So, we have to normalize.

i.e, $$P(Yes/Today) = \frac{P(Yes/Today)}{P(Yes/Today) + P(No/Today)}$$

$$= \frac{0.0212}{0.0212 + 0.0069}$$

$$= 0.7544$$

& $$P(No/Today) = \frac{P(No/Today)}{P(No/Today) + P(Yes/Today)} = \frac{0.0069}{0.0069 + 0.0212} = 0.2456$$

$\because P(Yes/Today) > P(No/Today)$. So, they can play golf in today's conditions.

⑧. Learning by Genetic Algorithm (GA):

Genetic Algorithms (GAs) are adaptive heuristic search algorithm that belongs to the larger part of evolutionary algorithms. It is based on the idea of neural selection & genetics. The historical data are provided to find out better solution or simply GA's are used to generate high quality solutions for optimization and search problem.

Genetic Algorithms (GA's) simulate the process of natural selection which means those species who can adopt to change their environment are able to survive & reproduce next generation (fittest survival). Each generation consists of a population of individuals and each individual represents a point in search space & possible solutions which are represented as string of characters/integers/floats/bits.

Operations in Genetic Algorithm:

i) Reproduction/Selection → This operator provides the performance to the individuals with good fitness score and allow them to pass their genes to the successive generator. To select the parents the techniques are round wheel, tournament selection, rank selection etc.

ii) Crossover → This represents mating between two individuals. Two individual are selected using selection operators and crossover sites are choosen randomly then genes at the crossover sites are exchanged thus, it creates completely new offsprings.

Example:

0	1	2	3	4	5	6	7	8	9

5	8	9	4	2	3	5	7	5	8

→

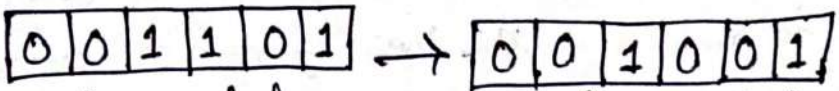
0	1	2	3	4	5	7	5	8

Child after crossover.

The popular crossovers are one-point crossover, multipoint crossover & uniform crossover.

III) Mutation → The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence.

Example:



Before mutation After mutation.

IV) Fitness function → It is the function which takes a candidate solution to the problem as input & produce output, determines how good or fit candidate is.

Genetic Algorithm:

- → Randomly initialize population.
- → Determine fitness of population.
- → Until convergence repeat:
  - → Select parents from population.
  - → Crossover and generate new population.
  - → Perform mutation on new population.
  - → Calculate fitness for new population.

Applications of GA's:

- → Recurrance Neural Networks.
- → Mutation Testing
- → Learning Fuzzy rule.

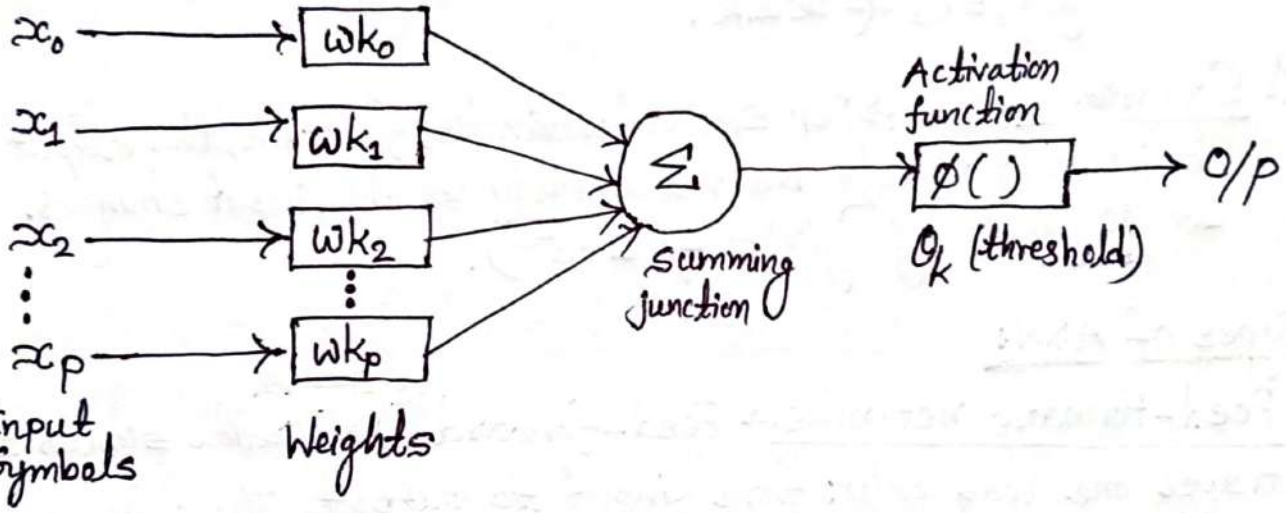
⊛ Learning with Neural Networks:

Artificial Neural Network (ANN):-

ANN is the non-linear parallely distributed & highly connected network having capacity of adaptivity, self-organization, fault tolerence & very large scale integration (VLSI) implementation of neurons. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological normal system. The network function is determined by connecting neurons in different levels & we can train them to perform some task by assigning some values to the connection called weight.

The goal of ANN is to solve problems in the same way that human mind does. ANN are widely used in computer vision, speech recognization, medical diagnosis, face recognization, signature verification etc.

⊛ Mathematical Model of ANN:



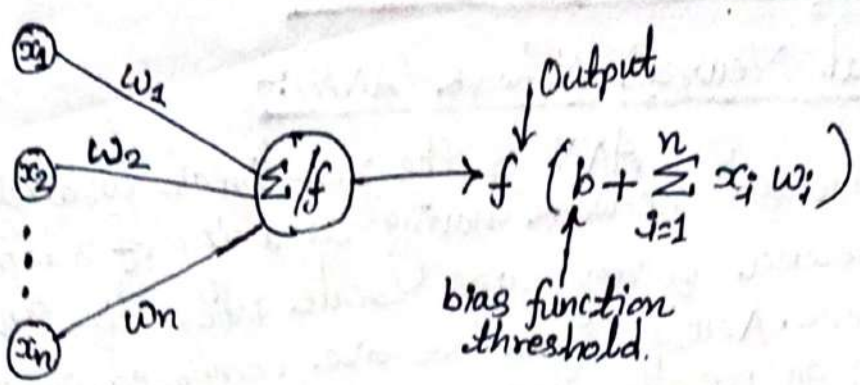
Inputs: Numerical values $x_1, x_2, x_3 \ldots x_p$ are the input in neurons.

Bias weight: Weight of all neurons determined by threshold (i.e, predefined value to make thing correct).

Activation function: The process of deciding output after aggregating input is called aggregation function. Each unit first computes a weighted sum of its inputs:

$in_i = \sum_{k=0}^{n} w_{k,i} \cdot x_k$, then it applies activation function g to this sum to derive the output: $O_i = g(in_i) = g\left(\sum_{k=0}^{n} w_{k,i} \cdot x_k\right)$.

where, g is activation function.



⊛ Types of Activation Functions:

i) Linear function → For linear activation functions, the output activity is proportional to the total weighted output. This function is given by $g(x) = kx + c$, where k & x are constant.

ii) Threshold function → For threshold functions, the output are set at one of the two levels, depending on whether the total input is greater than or less than some threshold value.

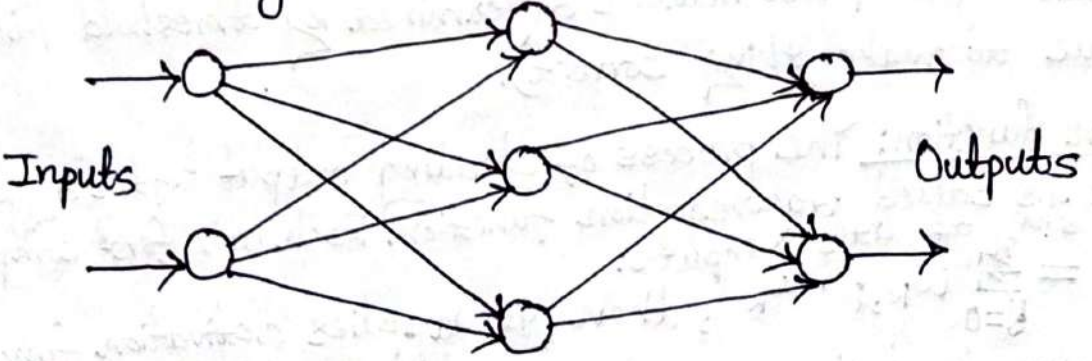
$g(x) = 1$, if $x >= k$

$g(x) = 0$, if $x < k$.

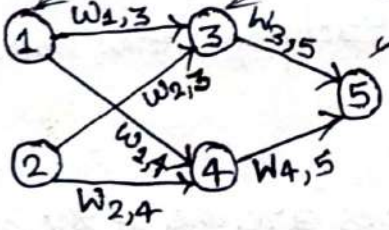
iii) Sigmoid function → For Sigmoid activation functions, the output varies continuously but not linearly as the input changes. It is given by $g(x) = 1/(1+e^{-x})$.

⊛ Types of ANN:

i) Feed-forward networks → Feed-forward ANNs allow signals to travel one way only: from input to output. There is no feedback (loops). i.e, the output of any layer does not affect the same layer. This types of ANNs are extensively used in pattern recognization.



Example:



first layer a_1
second layer a_2
third layer a_3

output की value हमें चाहिए

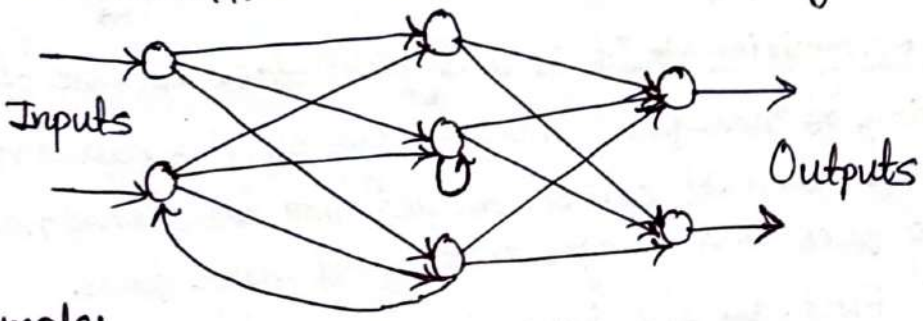
$$g_3 = g(w_{1,3} a_1 + w_{2,3} a_2)$$
$$g_4 = g(w_{1,4} a_1 + w_{2,4} a_2)$$
$$g_5 = g(w_{4,5} a_2 + w_{3,5} a_3)$$

Note: If fig. contains 2 layer i.e, a_1, a_2 then it is single layer. If contains more than two it is multilayer.

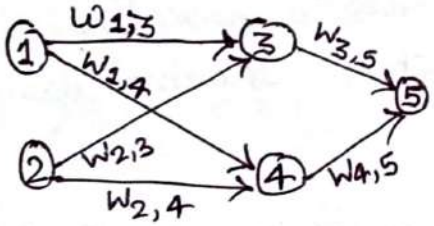
Examples described here are multilayer since contains more than 2 i.e, 3 layers

Feedback networks (Recurrent networks):

→ Feedback networks can have signals travelling in both directions by introducing loops in the network.



Example:



Here,

$$g_5 = g(w_{3,5} a_3 + w_{4,5} a_4)$$
$$= g(w_{3,5}\, g(w_{1,3} a_1 + w_{2,3} a_2) + g(w_{1,4} a_1 + w_{2,4} a_2)).$$

⊛ Applications of Artificial Neural Networks (ANNs):

Speech Recognition → Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. ANN has made great progress in this field. Following ANNs have been used for speech recognition:-

- → Multilayer networks
- → Multilayer networks with recurrent connections.
- → Kohonen self-organizing feature map.

Character Recognition → It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are the

Some ANNs which have been used for character recognition:

→ Multilayer neural networks such as Backpropagation neural networks.

→ Neocognitron.

iii) Signature Verification Application → Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique. The trained neural network will classify the signature as being genuine or forged under the verification stage.

iv) Human Face Recognition → It is a typical task because of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be processed. Then, the dimensionality of that image must be reduced. And at last it must be classified using neural network training algorithm.

⊛ Learning by Training ANN:

Learning means to adopt the changes in itself when there is change in environment. In complex system of ANN, after learning it changes its internal structure based on the information passing through it.

Learning is important in ANN because of some changes occuring in environment then ANN must change its inputs, outputs, activation function & Weight. To change the internal structure of ANN there are various methods (i.e, learning rules) which are as follows:

1) Hebbian Learning rule: Hebbian learning rule is introduced by Dnald Hebbian in 1949. It is also called unsupervised feed forward learning. This rule basically states that, if neuron i is near enough to excite neuron j and repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened and neuron j becomes more sensitive to

(i.e, made stronger)

from neuron i.

Hebb's law can be represented in the form of two rules:

i) If two neurons on either side of a connection are actived synchronously, then the weight of that connection is increased.

ii) If two neurons on either side of a connection are actived asynchronously, then the weight of that connection is decreased.

Hebb's Algorithm:

Step1: Initialize all weights and bias to 0.

Step2: Given a training input s, with it's target output t, set the activations of the input units: $x_i = s_i$

Step3: Set the activation of the output unit to the target value: $y = t$

Step4: Adjust the weights: $w_i(new) = w_i(old) + x_i y$

Step5: Adjust the bias (just like the weights): $b(new) = b(old) + y$

Problem: Construct a Hebb Net which performs like an AND function.

Solution:

Let us construct truth table of AND gate as follows:-

(b always 1 हुन्छ। x_1 र x_2 दुवै 1 भए मात्र y 1 हुन्छ अन्य −1, AND gate जस्तै same but 0 लाई −1 ले represent गरिएको छ)

INPUT				TARGET	
	x_1	x_2	b		y
X_1	−1	−1	1	Y_1	−1
X_2	−1	1	1	Y_2	−1
X_3	1	−1	1	Y_3	−1
X_4	1	1	1	Y_4	1

There are 4 training samples, so there will be 4 iterations.

Step1: Set weighs and bias to zero, $W = [0\ 0\ 0]$ and $b = 0$.

Step2: Set the activations of input units $x_i = s_i$ for $i = 1$ to 4.

$X_1 = [-1\ -1\ 1]$ (i.e. $X_1 = [x_1\ x_2\ b]$)

$X_2 = [-1\ 1\ 1]$

$X_3 = [1\ -1\ 1]$

$X_4 = [1\ 1\ 1]$

Step3: Output value is set to $y=t$.

Step4: Adjust (or Modify) weights using Hebbian Rule:

1st iteration: $W(new) = w(old) + x_1 y_1$

$= [0\ 0\ 0] + [-1\ -1\ 1] . [-1]$

$= [1\ 1\ -1]$

2nd iteration: $W(new) = w(old) + x_2 y_2$

(for 2nd iteration, final wt. of 1st iteration will be used & so on...)

$= [1\ 1\ -1] + [-1\ 1\ 1] . [-1]$

$= [2\ 0\ -2]$

3rd iteration: $w(new) = w(old) + x_3 y_3$

$= [2\ 0\ -2] + [1\ -1\ 1] . [-1]$

$= [1\ 1\ -3]$

4th iteration: $w(new) = w(old) + x_4 y_4$

$= [1\ 1\ -3] + [1\ 1\ 1] . [1]$

$= [2\ 2\ -2]$.

So, the final weight matrix is $[2\ 2\ -2]$.

($[x_1\ x_2\ b]$ form)

Testing the Network: (Step 5)

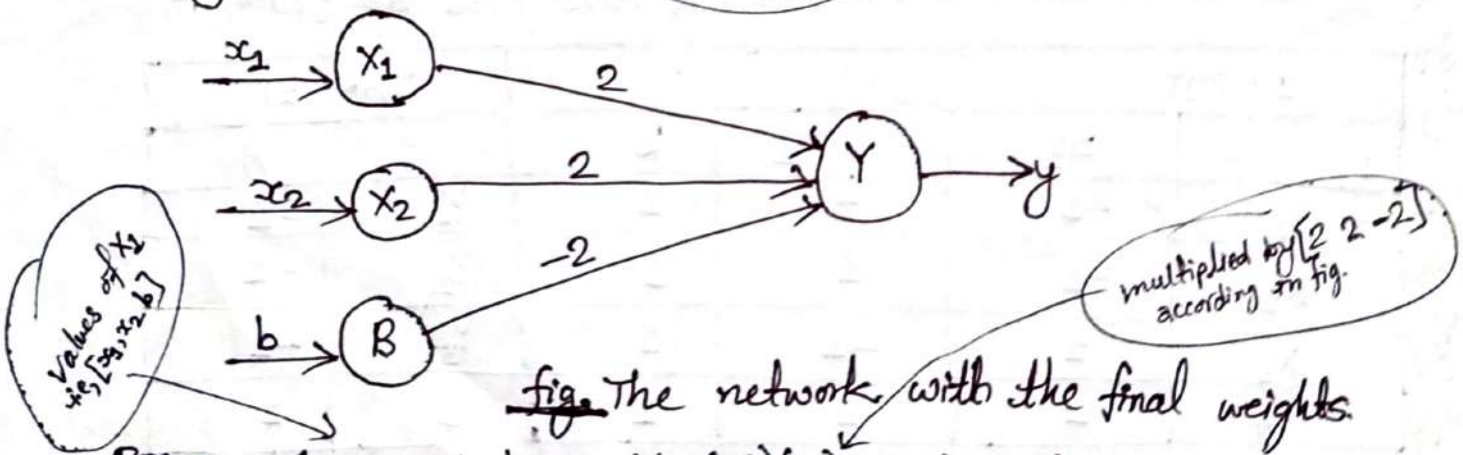


fig. The network with the final weights.

For $x_1=-1, x_2=-1, b=1$, $Y=(-1)(2)+(-1)(2)+(1)(-2)=-6$

For $x_1=-1, x_2=1, b=1$, $Y=(-1)(2)+(1)(2)+(1)(-2)=-2$

For $x_1=1, x_2=-1, b=1$, $Y=(1)(2)+(-1)(2)+(1)(-2)=-2$

For $x_1=1, x_2=1, b=1$, $Y=(1)(2)+(1)(2)+(1)(-2)=2$

(i.e, first 3 values negative and final only positive)

The results are all compatible with the original table.

Decision Boundry:

We have $2x_1+2x_2-2b=y$ (from fig)

Now replacing y with 0, we get, $2x_1+2x_2-2b=0$

Since, bias, $b=1$, so, $2x_1+2x_2-2(1)=0$ i.e, $2(x_1+x_2)=2$

From $2(x_1+x_2)=2$, we get final equation, $x_2=-x_1+1$.

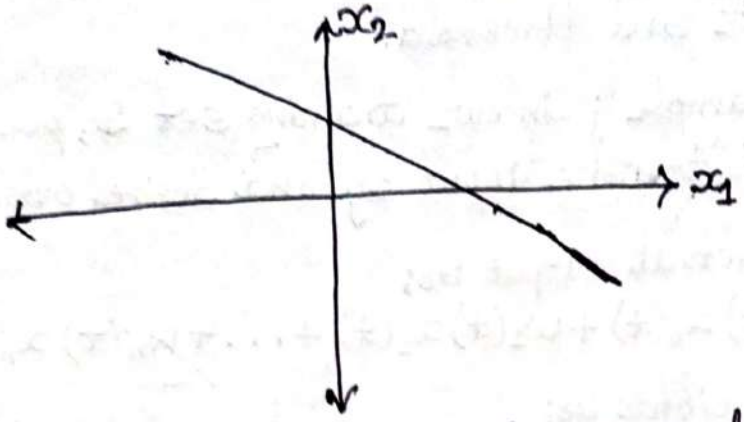


fig. Decision Boundry of AND Function.

less imp but once have a look similar to algorithm only use of { } braces

Pseudocode for Hebb's Net (Supervised)

```
Initialization; for i=1 to n : {b=0, w_i=0}.
For each of the training samples s: to do
{ /*s is the input pattern, t the target output of the sample */
   for i=1 to n {x_i=s_i}   /*set to input units*/
                y=t         /*set y to the target*/
   for i=1 to n{
                w_i=w_i+x_i*y
                } /*update weight*/
     b=b+x_i*y  /*update bias*/
}
```

2) Perceptron Learning:

Perceptron learning is used in supervised learning rule to classify the data in two classes. It consist of a single neuron with arbitary value 1 or 0 depending on threshold & also consist of bias weight. Training patterns are presented to the network's inputs; the output is computed.

Variables used:

- $y=f(x)$ denotes the output for an input vector x.
- $D=(x_1,d_1),(x_2,d_2),\ldots,(x_n,d_n)$ is the training set of n samples.

where, $x_j \rightarrow$ input vector
$d_j \rightarrow$ desired output for x_j.

Algorithm:

Step1: Initialize weights and threshold.

Step2: For each example j in our training set D, perform following steps 3 & 4 over the input x_j and desired output d_j.

Step3: Calculate the actual output as;

$$y(t) = g[w_0(t)\,x_0(t) + w_1(t)\,x_1(t) + \ldots + w_n(t)\,x_n(t)]$$

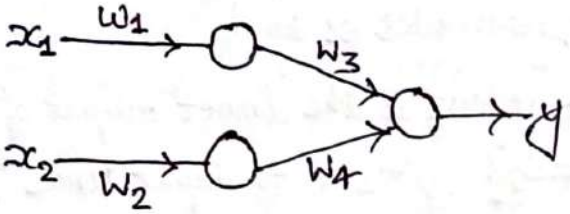
i.e, $y(t) = g\left[\sum_i w_i(t)\,x_i(t)\right]$

Step4: Update the weights as;

$$w_i(t+1) = w_i(t) + \alpha\,[d(t) - y(t)]\,x_i(t)$$

where, $0 <= \alpha <= 1$ (learning rate)

Example:



Given,

desired output $d(t)$.

Actual output is computed as;

$$y(t) = g(y_1 w_3(t) + y_2 w_4(t))$$

where, $y_1 = g(x_1 w_1(t))$

$y_2 = g(x_2 w_2(t))$

Now update weight as;

$$w_1(t+1) = w_1(t) + \alpha[d(t) - y(t)]\,x_1(t)$$
$$w_2(t+1) = w_2(t) + \alpha\,[d(t) - y(t)]\,x_2(t)$$
$$w_3(t+1) = w_3(t) + \alpha\,[d(t) - y(t)]\,y_1(t)$$
$$w_4(t+1) = w_4(t) + \alpha\,[d(t) - y(t)]\,y_2(t)$$

After updating weights, we recompute $y(t)$. If the desired solution is obtained then stop otherwise we will iterate again.

3) Back-propagation learning:-

It is a supervised learning method, and is an implementation of Delta rule. It requires a teacher that can calculate the desired output for any given input. It is most useful for feed-forward networks. Back propagation requires that the activation function used by the artificial neurons (or nodes) is differentiable.

Algorithm:

1. Randomly choose the initial weight.
2. Compute output (y) from neural network:

$$y = g(\sum_i x_i w_i) \text{ where, } g(x) = \frac{1}{1+e^{-x}} \text{ is sigmoid function.}$$

3. Compute error $\delta = t - y$; where t is targated output and y is actual output.
4. Propagate this δ error back to layer of neural network and compute error at each layer as;

$$\delta_i = \sum_i w_{ij} \delta_j$$

5. Update the weights as;

$$w_{ij}(new) = w_{ij}(old) + \alpha \delta_j \frac{d(y_i)}{de} x_i$$

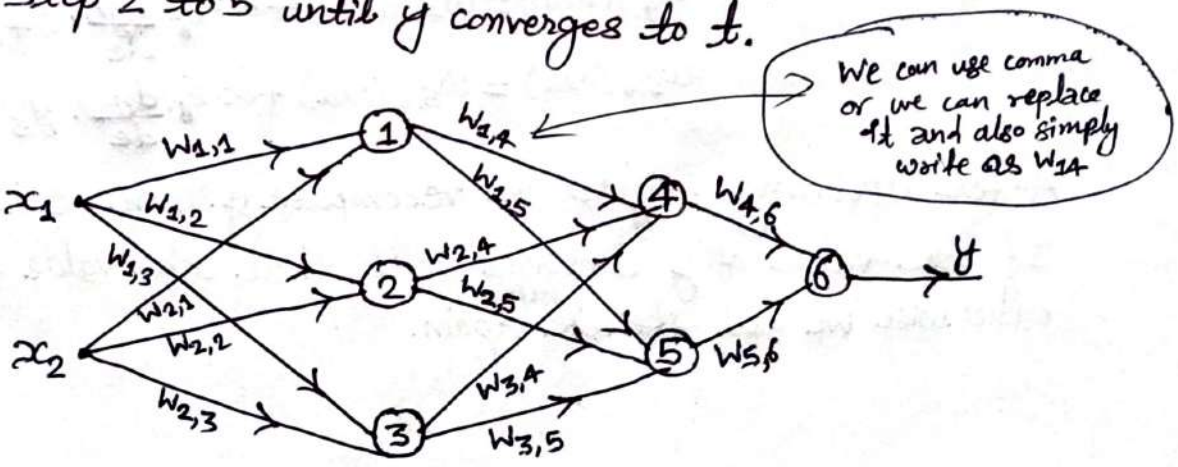
where, $\frac{d(y_i)}{de}$ is derivative of y_i w.r.t. e.

α is learning rate $0 \le \alpha \le 1$

x_i is input to node.

6. Repeat step 2 to 5 until y converges to t.

Example 1:



$$y = g(y_4 w_{4,6} + y_5 w_{5,6})$$

where, $y_4 = g(y_1 w_{1,4} + y_2 w_{2,4} + y_3 w_{3,4})$

$y_5 = g(y_1 w_{1,5} + y_2 w_{2,5} + y_3 w_{3,5})$

where, $y_1 = g(x_1 w_{1,1} + x_2 w_{2,1})$

$y_2 = g(x_1 w_{1,2} + x_2 w_{2,2})$

$y_3 = g(x_1 w_{1,3} + x_2 w_{2,3})$

Given target t, Compute $\delta = t - y$.

Now calculate backpropagate error as;

$\delta_4 = w_{4,6} * \delta$

$\delta_5 = w_{5,6} * \delta$

$\delta_1 = w_{1,4}\delta_4 + w_{1,5}\delta_5$

$\delta_2 = w_{2,4}\delta_4 + w_{2,5}\cdot\delta_5$

$\delta_3 = w_{3,4}\cdot\delta_4 + w_{3,5}\cdot\delta_5$

if confusion once look and compare with algorithm.

Now update weight as;

$w_{1,1}(new) = w_{1,1}(old) + \alpha\,\delta_1 \cdot \frac{d(y_1)}{de} \cdot x_1$

$w_{1,2}(new) = w_{1,2}(old) + \alpha\,\delta_2 \frac{d(y_2)}{de} \cdot x_1$

$w_{1,3}(new) = \ldots$ (similarly)

$w_{2,1}(new) = w_{2,1}(old) + \alpha\,\delta_1 \cdot \frac{d(y_1)}{de} \cdot x_2$

$\vdots$

$w_{2,5}(new) = w_{2,5}(old) + \alpha\,\delta_5 \cdot \frac{d(y_5)}{de} \cdot y_2$

$w_{3,4}(new) = w_{3,4}(old) + \alpha\,\delta_4 \cdot \frac{d(y_4)}{de} \cdot y_3$

$\vdots$

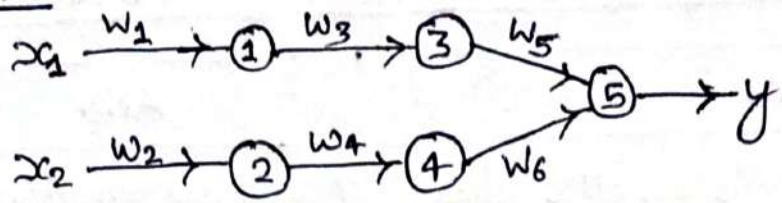
$w_{4,6}(new) = w_{4,6}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_4$

$w_{5,6}(new) = w_{5,6}(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_5$

$w_{1,3}$ हमको दिया है, As $w_{1,1}$, $w_{1,2}$ similarly we can do $w_{1,3}$ easily do yourself

After updating weights., we recompute y. Then we compute $\delta = t - y$. If the value of y converges to 't' then the value is fixed up otherwise we will iterate again.

Example 2:



$$y = g(y_3 w_5 + y_4 w_6)$$

where,
$$y_3 = g(y_1 w_3)$$
$$y_4 = g(y_2 w_4)$$

where,
$$y_1 = g(x_1 w_1)$$
$$y_2 = g(x_2 w_2)$$

given target t, Compute $\delta = t - y$.

Now compute backpropagate error as;

$$\delta_3 = w_5 \cdot \delta$$
$$\delta_4 = w_6 \cdot \delta$$
$$\delta_1 = w_3 \cdot \delta_3$$
$$\delta_2 = w_4 \cdot \delta_4$$

Now update weight as;

$$w_5(new) = w_5(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_3$$
$$w_6(new) = w_6(old) + \alpha \cdot \delta \cdot \frac{dy}{de} \cdot y_4$$
$$w_3(new) = w_3(old) + \alpha \cdot \delta_3 \cdot \frac{d(y_3)}{de} \cdot y_1$$
$$w_4(new) = w_4(old) + \alpha \cdot \delta_4 \cdot \frac{d(y_4)}{de} \cdot y_2$$
$$w_1(new) = w_1(old) + \alpha \cdot \delta_1 \cdot \frac{d(y_1)}{de} \cdot x_1$$
$$w_2(new) = w_2(old) + \alpha \cdot \delta_2 \cdot \frac{d(y_2)}{de} \cdot x_2$$

After updating weights, we recompute y. Then we compute $\delta = t - y$. If the value of y converges to t then the value is fixed up otherwise we will iterate again.

⊛ Biological Neural Networks (BNN) vs. Artificial Neural Networks (ANN):

Criteria	BNN	ANN
i) Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
ii) Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes (mainly depends on type of application and network designer).
iii) learning	They can tolerate ambiguity.	Very precise, structured and formatted data is required to tolerate ambiguity
iv) Fault tolerance	Performance degrades with even partial damage.	It is capable of robust performance, hence has the potential to be fault tolerent.
v) Storage Capacity	Stores the information in the synapse.	Stores the information in continious memory locations.

⊛ Supervised Learning vs. Unsupervised Learning:

Supervised Learning	Unsupervised learning
i) Supervised learning algorithms are ~~used~~ trained using labeled data.	i) Unsupervised learning algorithms are trained using unlabeled data.
ii) Supervised learning model predicts the output.	ii) Unsupervised learning model finds the hidden patterns in data.
iii) In supervised learning, input data is provided to the model along with the output.	iii) In unsupervised learning, only input data is provided to the model.
iv) Supervised learning needs supervision to train the model.	iv) Unsupervised learning does not need any supervision to train the model.
v) Supervised learning model produces an accurate result.	v) Unsupervised learning model may give less accurate result as compared to supervised learning.

UNIT-6

Applications of AI

Heuristic meaning informative or instructional

⊛. 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 it's knowledge base using the reasoning and inference rules according to the user queries. The system helps in decision making for complex problems using both facts and heuristics like a human expert. The performance of an expert system is based on the expert's knowledge stored in it's knowledge base.

Expert systems are used by most of the larger or medium sized organization as a major tool for improving productivity and quality. These systems are designed for a specific domain, such as medicine, science etc. One of the common example of expert system is a suggestion of spelling errors while typing in the Google search box. Expert systems hold the characteristics like high performance, understandable, reliable, and highly responsive.

⊛. Components of Expert System:

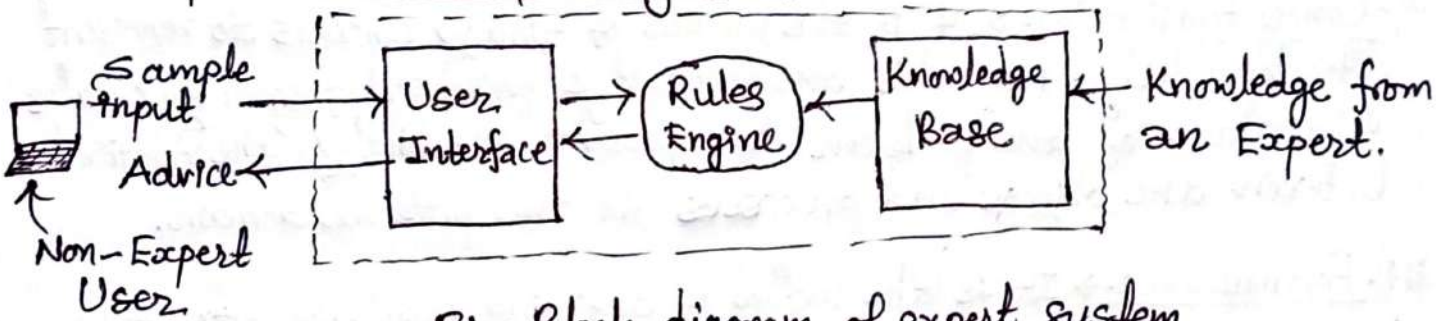


Fig: Block diagram of expert system.

An expert system mainly consists of following three components:

1) User Interface → With the help of user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine (or rules engine). After getting response from the inference engine, it displays the output to the user.

ii) Inference Engine (Rules of Engine) → The inference engine is known as brain of expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.

iii) Knowledge Base → The knowledge base is a type of storage that stores knowledge acquired from different experts of the particular domain. The more the knowledge base, the more precise will be the expert system. It is similar to a database that contains information and rules of a particular domain or subject.

⊛. Development of Expert Systems:

An expert system typically is developed and refined over a period of several years. Following are the stages of expert system development that provide us some insight into the ways in which expert systems are developed.

i) Identification → It is the method of determining the characteristics of the problem. It helps to determine the exact nature of the problem instead to feel that the system would be helpful in certain situation.

ii) Conceptualization → It is the method of finding concepts to represent the knowledge. For this the knowledge engineer frequently creates a diagram of the problem to represent graphically, the relationship between the objects and processes in the problem domain.

iii) Formalization → It is the method of designing knowledge structure using knowledge representation techniques. During formalization, it is important that the knowledge engineer be familiar with the various techniques of the knowledge representation and the expert system tools.

iv) Implementation → It is the method of creating prototypes of expert system. Many scientists actually consider the first prototype to be a "throw-away" system, useful for evaluating progress but hardly a usable expert system.

v) Testing → It is the method of validating the implemented expert system. Testing provides opportunities to identify the weakness in the structure and implementation of the system and to make the appropriate corrections.

⊛. Features of an Expert System:

→ It should be able to respond to simple questions.
→ It should be able to learn new knowledge.
→ It should be easily modified.
→ It should be adaptive and flexible.
→ It should be able to explain its advice.
→ It should be goal oriented.

Q. Explain knowledge engineering with a block diagram.

Ans: Knowledge engineering is a field of AI that creates rules to apply data in order to imitate the thought process of human expert. Knowledge engineering attempts to take challanges and solve problems that would usually require a high level of human expertise to solve.

In general, knowledge engineering is the process of understanding and representing a human knowledge in a computer as a program. Knowledge engineering includes:

i) Knowledge acquisition.
ii) Knowledge representation.
iii) Inferencing.
iv) Explanation and justification.

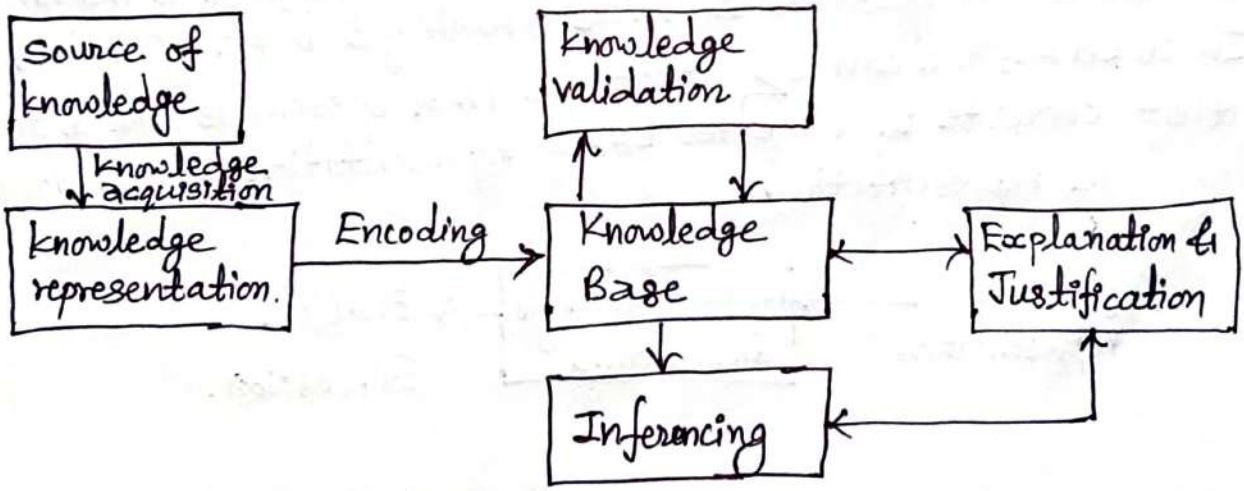
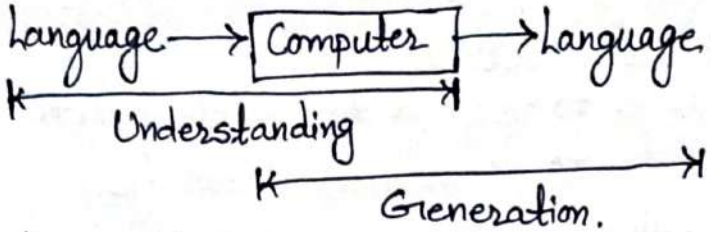


Fig: Knowledge Engineering process.

⊛ Natural Language Processing (NLP):

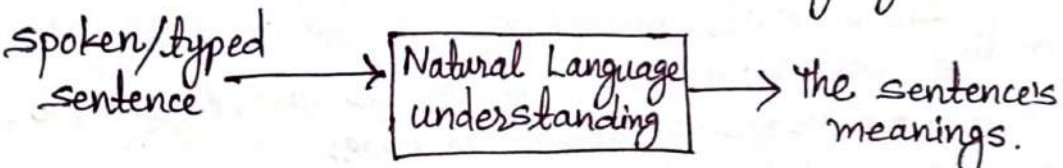
Natural language processing is a technology which involves converting spoken or written language into a form which can be processed by computers and vice-versa.



Voice recognition software, Text-to-speech synthesizers, Grammer checkers, Machine translation systems etc. are some of the better-known applications of NLP language. NLP is composed of two parts: NLU (Natural language understanding) and NLG (Natural language generation).

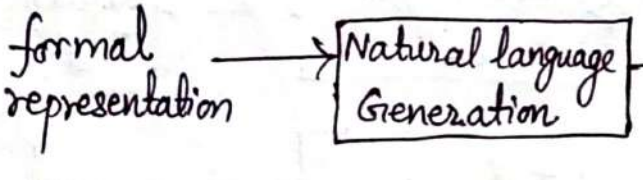
⊛ Natural Language Understanding (NLU):

It is the process of mapping the given inputs in natural language into useful representation and analyzing different aspects of the language. Developing programs that understand a natural language is a difficult problem. Natural languages are large.



⊛ Natural Language Generation (NLG):

It is the process of producing meaningful phrases and sentences in the form of natural language from a machine representation system such as knowledge base or logical form. In a sense, one can say that an NLG system is like a translator that converts a computer based representation into a natural language representation.

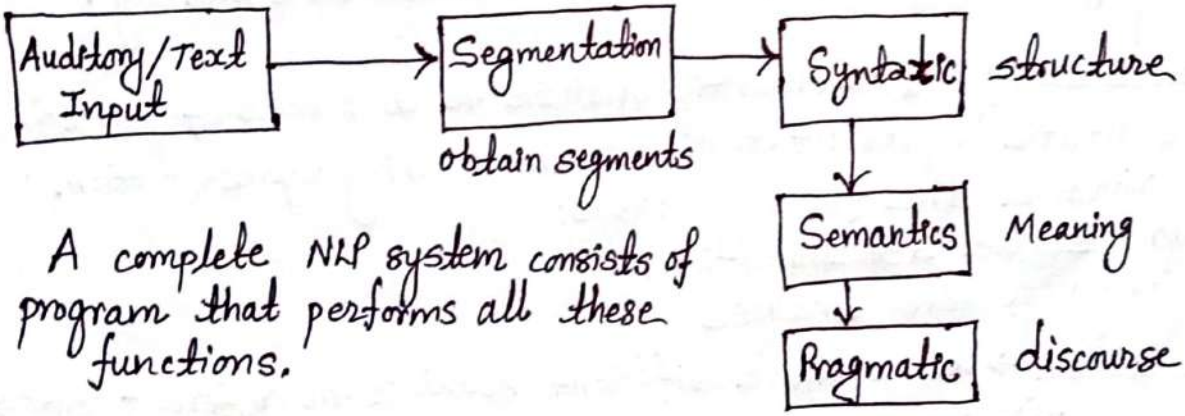


⊛. NLU vs. NLG:

NLU	NLG
i) NLU is taking some spoken/typed sentence and working out what it means.	i) NLG is taking some formal representation of what we want to say & working out a way to express it in a natural language.
ii) In NLU the system needs to disambiguate the input sentence to produce the machine representation language.	ii) In NLG the system needs to make decisions about how to put a concept into words.
iii) Different levels of analysis required: morpological analysis, syntatic analysis, semantic analysis.	iii) Different levels of synthesis required: deep learning, syntatic generation.
iv) NLU is most harder than NLG.	iv) NLG is less harder than NLU.

⊛ Steps of Natural Language Processing: (OR Parameters in NLP)

(or Process)



A complete NLP system consists of program that performs all these functions.

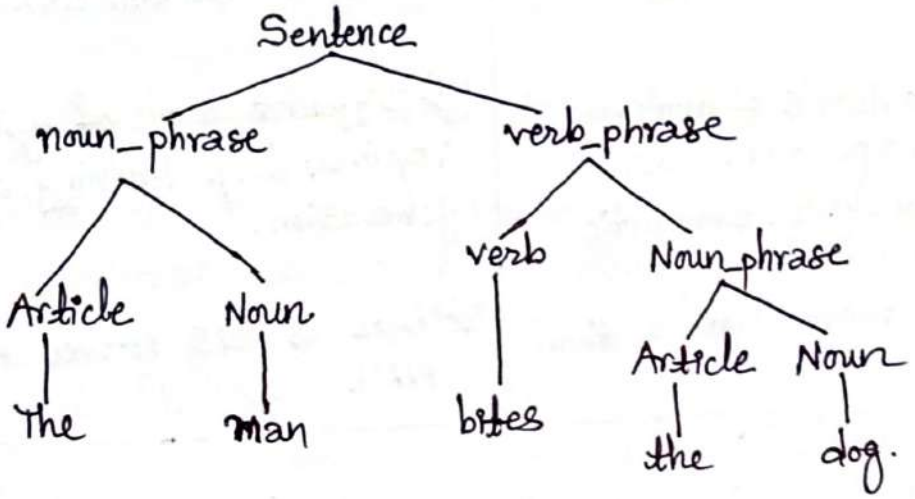
i) Input/source → The input of a NLP system can be written text or speech. Quality of input decides the possible errors in language processing that is high quality input leads to correct language understanding.

ii) Segmentation → The text inputs are divided into segments (chunks) and the meaning of individual segments are analyzed.

→ Syntatic Analysis → Syntatic analysis takes an input sentence and produces a representation of its grammatical structure. A grammar describes the valid parts of speech of a language and how to combine them into phrases.

A computer grammar specifies which sentences are in a language and their parase tree. A parase tree is a hierarchical structure that shows how the grammar applies to the input. Each level of the tree corresponds to the application of one grammar rule.

Example: Parse tree



Fig; parse tree for the sentence "The man bites the dog".

→ Semantic Analysis → Semantic analysis is a process of converting the syntatic representations into a meaning representation. This involves the following tasks:

→ word sense determination.
→ sentence level analysis.

Word sense → Words have different meanings in different contexts.

Example: Susmita had a bat in her office.

bat = "a baseball thing"
bat = "a flying mammal".

sentence level analysis → Once the words are understood, the sentence must be assigned some meaning.

Example: I saw an astronomer with a telescope.

v) Pragmatic Analysis → It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected. The main focus is on what was said is reinterpreted on what it actually means.

Morphology → It is the process of recognizing the suffixes and prefixes that have been attached to a word.
For example: adjective + ly → adverb [e.g, Friend + ly = friendly].

⊛. Importance of NLP:

→ NLP helps to make communication easier between the user and computer system.
→ It helps to understand large social data available in the internet.
→ It improves the efficiency and accuracy of documentation and identify the most relevent information from large database.

⊛. Machine Translation:-

The term "machine translation" (MT) is used in the sense of translation of one language to other. The ideal aim of machine translation system is to produce the best possible translation without human assistance. An example of machine translation is "Google Translator" which can translate english language to other languages like nepali, hindi and vice-versa.

The translation quality of the machine translation systems can be improved by pre-editing and post-editing in MT. Pre-editing means adjusting the input by making prefixes, suffixes, clause boundries etc. Post-editing means controlling the vocabulary to the output of the MT.

⊛ Types of machine translations:

There are four types of machine translations which are as follows:

i) Rule based machine translation (RBMT) → It translates on the basis of grammatical rules. It conducts a grammatical analysis of the source language and the target language to generate the translated sentence. It can translate the source language

directly to the target language.

ii) Statistical machine translation (SMT) → It offers good solution to ambiguity problem. SMT are robust and work well even if there are errors and the presence of new data. SMT aims to determine the correspondence between a word from the source language and a word from the target language.

iii) Hybrid machine translation (HMT) → It is the blend of RBMT and SMT. It holds a translation memory, making it far more effective in terms of quality. However, even HMT has its drawbacks, the main drawback is the need for extensive editing.

iv) Neural machine translation (NMT) → It depends on neural network models (based on human brain) to develop statical models for the purpose of translation. The primary benefit of NMT is that it provides a single system that can be trained to decode the source and target text.

⊛. Machine Vision Concepts :-

Machine vision is the ability of a computer to "see". A machine vision system employs one or more video cameras, analog-to-digital conversion, and digital signal processing. The resulting data goes to a computer or robot controller. It uses different components to visually analyze an operation or activity.

Two important specifications in any vision system are the sensitivity and the resolution. Sensitivity is the ability of a machine to see in dim light, or to detect weak impulses at invisible wavelengths. Resolution is the extent to which a machine can differentiate between objects. Machine vision systems have two primary hardware elements the camera, which serves as the eye of the system and a computer video analyzer.

Components: A typical machine vision system will consist of the following components:

→ One or more digital or analog cameras with suitable optics for acquiring images, such as lenses to focus the desired field of view. It also consists image sensor which is responsible for analysing captured images or presence of defects.

→ Input/Output hardware (e.g, digital I/O) or communication links (e.g, network connection or RS-232) to report result.

→ A synchronizing sensor for part detection to trigger image acquisition and processing and some form of actuators to sort, route or reject defective parts.

→ A program to process images and detect relevent features.

Applications:

→ Electronic component analysis
→ Signature identification.
→ Optical character recognition.
→ Handwriting recognition.
→ Object recognization
→ Pattern recognization.
→ Materials inspection.
→ Medical image analysis.

⊛. Robotics:

Robotics is a branch of engineering and science that includes electronics engineering, mechanical engineering, computer science and so on. This branch deals with the design, construction, sensory feedback and information processing. These robots are designed to be for any purpose like bomb detection, industrial use and many more. Robots can take any form but many of them have given the human appearance.

The advantage of using robots is they can get information that a human can't. They can perform tasks without any mistake and efficiently as well as fast. The disadvantage of using robot is that people working in factories may loose their jobs and they need high maintenance.

Robot Hardware:- A robot hardware generally consists of 5 basic components as follows:

i) **Controller** → Every robot is connected to a computer controller, which regulates the components of the arm and keeps them working together. Almost all robots are pre-programmed but in future controllers with AI could allow robots to think on their own, even program themselves.

ii) **Arm** → The arm is the part of the robot that positions the end-effector and sensors to do their pre-programmed business. Many are built to resemble human arms and work like human arms.

iii) **Drive** → The links (the sections between the joints) are moved into their desired position by the drive. Typically a drive is powered by hydraulic pressure or electricity.

iv) **End-Effector** → The end-effector could be thought of as the "hand" on the end of robotic arm. There are many possible end-effectors like gripper, vaccum pump, welder, spray gun etc. that help it to do its job.

v) **Sensor** → The sensors give the robot controller information about its surroundings and lets it to know the exact position of the arm, or the state of world around it. Robot sensors can detect infrared radiation to "see" in the dark.

Robotic Perceptions:- Robotic perception is related to many applications in robotics where sensory data and artificial intelligence/machine learning techniques are involved. Examples of such applications are object detection, environment representation, scene understanding, activity recognition etc. It contains the algorithms and techniques that empower robots ~~from~~ to learn from sensory data and, based on learned models, to react and take decisions accordingly. Robotic perception systems are evolving in a way that new applications and tasks are becoming a reality.