Page No.
Date

# Artificial Intelligence Notes
ICT 7th Semester

by


## Mukesh Singh
(ICT 5th batch, 2073)

Sukuna Multiple Campus

Sundarharaincha-12, Morang

| Units | Chapter Name / Contents | Page No. |
|---|---|---|
| Unit 1 | Introduction to AI (10)<br>1.1 Introduction to Artificial Intelligence<br>1.2 Brief History of AI<br>1.3 Applications of Artificial Intelligence<br>1.4 Definition and Importance of Knowledge,<br>1.5 Learning Agent and importance measure<br>1.6 Problems Definition, Real life problems and well-defined Problem | |
| Unit-2 | Search Techniques (20)<br>2.1 Uninformed search techniques: depth first search, breadth first search, depth limit search, Iterative deepening search,<br>2.2. Heuristics search techniques: Greedy Best first search, A* search, Hill Climbing, Game playing, Adversial Search techniques - mini-max procedure, alpha beta pruning | |
| Unit-3 | Knowledge, Reasoning & Planning (20)<br>3.1. formal logic-connectives: truth tables, syntax, semantics, tautology, validity, well formed-formula,<br>3.2. Propositional logic, Inference with PL: Resolution, Backward Chaining and Forward Chaining | |

3.3. First Order Predicate Logic (FOPL), quantification, inference with FOPL; By Converting into PL (Existential and Universal instantiation), Directly with FOPL (Unification and lifting, resolution backward chaining, forward chaining).

3.4. Rules based deduction system,

3.5. Statistical Reasoning - Probability and Bayes' theorem and causal Network, reasoning in belief network

Unit-4 Structured Knowledge Representation

4.1 Representation and mappings
4.2 Approaches to knowledge Representation, Issues in knowledge Representation.
4.3. Semantic nets, frames,
4.4 Conceptual dependencies and scripts.

Unit-5 Application of AI system in education (20)

5.1. Expert Systems (Architecture, Expert System development process),
5.2 Application of Expert system in education
5.3. Neural Network (Mathematical Model, get realization Network structure)
5.4 Application of Neural network in education

5.5 Natural Language Processing (Steps of NLP, parsing)
5.6 Application of NLP in education
5.7 Basic Concepts of Machine Learning and Visioning
5.8 Application of Machine Learning in education

# Unit-1

# Introduction to AI

Ajanta

Page No.
Date

## 1.1. Introduction to Artificial Intelligence

Artificial Intelligence is composed of two words Artificial and Intelligence, where Artificial defines "thinking power", hence AI means "a man-made thinking power."

Definition of AI:

It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions.

Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems.

## 1.2 Brief History of AI

Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.

### Maturation of AI (1943-1952)

- 1943 ⇒ Evolution of artificial neurons by McCullach and fits.
- 1950 → Alan Turing publishes "Computing Machinery and Intelligence".

### The birth of AI (1952-1956)

- 1955 → Allen Newell and Herbert A. Simon created the "first AI program, Logic Theorist.
- 1956 → The word Artificial Intelligence first adopted by John McCarthy at the Dartmouth Conference.

### The golden years - Early enthusiasm (1956-1974)

- 1966 → Joseph Weizenbaum created the first chatbot in 1966, which was named ELIZA.
- 1972 → The first intelligent humanoid robot was

named WABOT-1 was built in Japan.

## The first AI winter (1974-1980)

- AI winter refers to the time period where Computer scientist dealt with a severe shortage of funding from government for AI researches.
  Computers became faster and affordable.

## A boom of AI (1980-1987)

- 1980 → The first national conference of the American Association of Artificial Intelligence was held at Stanford University.

## The Second AI winter (1987-1993)

- Again investors and government stopped funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

## The emergence of intelligent agents (1993-2011)

- 1997 → IBM Deep Blue became the first computer to beat a world chess champion, Gary Kasparov.

- 2002 → AI entered the home in the form of Roomba, a vacuum cleaner.

- 2006 → AI came in the Business world till 2006.

Companies like Facebook, Twitter, and Netflix also started using AI.

Deep learning, big data and artificial general intelligence. (2011 - present)

- 2011 → IBM's 'Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles.

- 2012 → Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.

- 2014 → Chatbot "Eugene Goostman", won a competition in the infamous "Turing test".

- 2018 → The "Project Debator" from IBM debated on complex topics with two master debators and also performed extremely well.

## 1.3 Applications of Artificial Intelligence

Artificial Intelligence has various applications in today's society. following are some sectors which have the application of AI:

1. AI in Astronomy → AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare → Healthcare industries are applying AI to make a better and faster diagnosis than humans.

3. AI in Gaming → The AI machines can play strategic games like chess, where the machine needs to think of a large number of places.

4. AI in finance → The finance industry is using automation, chatbot, adaptive intelligence and machine learning into financial processes.

5. AI in Data Security → AI can be used to make your data more safe and secure.

6. AI in Social Media → AI can organize and manage massive amount of data of social media sites.

7. AI in Travel & Transport → Travel industries are using AI powered chatbots which can make human

like interaction with customers for better and fast response.

8. AI in Robotics → The intelligent Humanoid named as Erica and Sophia has been developed which can talk and behave like humans.

9. AI in Entertainment → We are currently using some AI based application in our daily life with some entertainment services such as Youtube, Netflix or Amazon.

10. AI in education → AI chatbot can Communicate with students as a teaching assistant.

## 1.4. Definition and Importance of Knowledge

### What is knowledge representation?

- Knowledge representation and reasoning (KR, KRR) is the part of AI which concerned with AI agents thinking ~~how~~ and how thinking contributes to intelligent behaviour of agents.

- It is also a way which describes how we can represent knowledge in AI. Knowledge representation is not just storing data into database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

The different kinds of knowledge that need to be represented in AI include:

- Objects
- Events
- Performance
- Facts
- Meta-knowledge
- Knowledge-base

### Different types of knowledge:

1. Declarative Knowledge → It includes concepts, facts, and objects and expressed in a declarative sentence.

2. Structural knowledge → It is a basic problem-solving

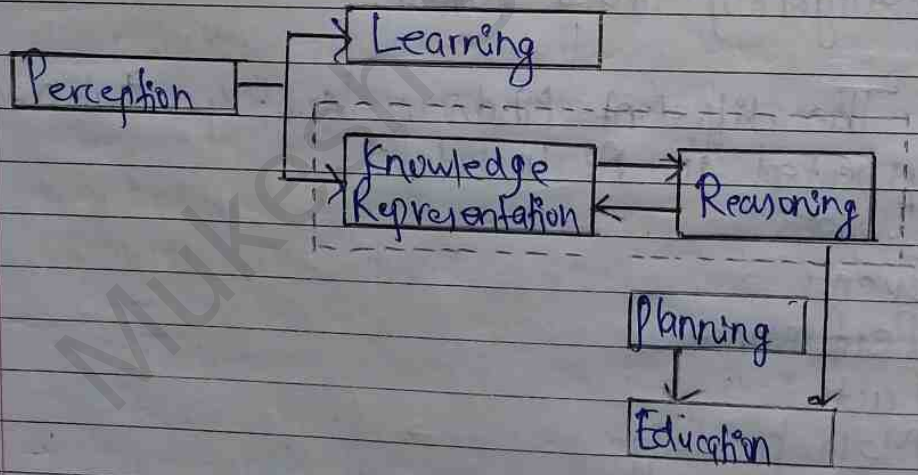
knowledge that describes the relationship between concepts and objects.

3. Procedural Knowledge → This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.

4. Meta-Knowledge → Meta knowledge defines knowledge about other types of knowledge.

5. Heuristic-Knowledge → This represents some expert knowledge in the field or subject.

# AI Knowledge Cycle:




The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence.

An AI system has following components for displaying intelligent behaviour:

i) Perception → AI system has perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input.

ii) Learning → The learning component is responsible for learning from data captured by perception component.

iii) Knowledge Representation and Reasoning → The main Component in the cycle is Knowledge Representation and Reasoning which shows human like intelligence in the machines. These two components are independent with each other but also coupled together.

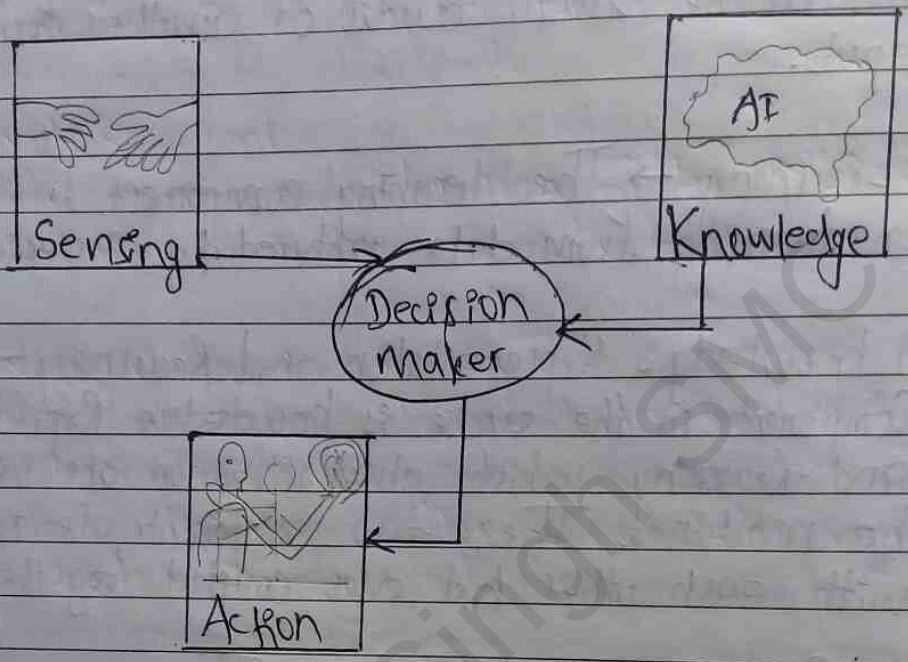
iv) Planning and Execution → The planning and execution components depend on the analysis of knowledge representation and reasoning.

# The relation between knowledge and intelligence

Knowledge of real-world plays a vital role in intelligence and same for creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behaviour in AI agents.

Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behaviour of that input.

Let's take an example to understand the relationship:




As we can see in the above diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behaviour.

Importance of knowledge in artificial intelligence:

→ to search through large amounts of information in order to find important data.

→ to find patterns that were previously unknown.

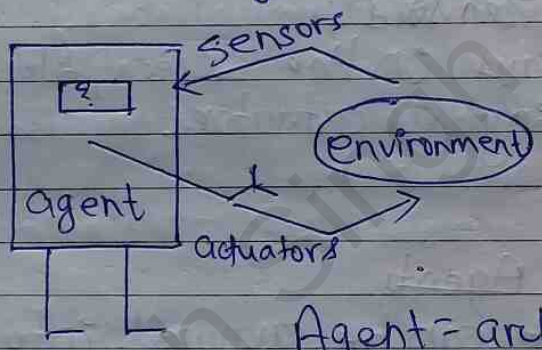
→ to save a number of problems.

## 1.5 Learning agents and its performance measure

### Agents

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

Note: Every agent can perceive its own actions (but not always the effects).




Agent = architecture + agent program

Sensor → Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

Actuators → Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.

An agent runs in the cycle of perceiving, thinking and acting. An agent can be:

- Human agent →
→ Has eyes, ears and other organs for sensors;
→ Has legs, mouth, hand etc. for actuators.

- Robotic agent
→ Has cameras, infrared range finders, NLP for sensors
→ Has various motors for actuators.

- Software agent
→ Has keystrokes, file contents, received network packages as sensors;
→ And displays on the screen, files, sent network packets acting as actuators.

## Types of Agents

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

1. Simple reflex agents.
These agents simply decide actions based on their current percept. By identifying that certain actions are warranted in certain conditions the agent can build a list of condition-actions rules and used them to decide which actions to take.

2. Model based reflex agents
Here, the agent maintains a model of the world that include an internal state, knowledge

about how the world evolves ('laws of nature' from the perspective of the agent) and knowledge about how the agent's actions affect the world. They can help with partially observable worlds.

3. Global based agents
The agent is given a goal and hence the agent can now modify its other aspects as necessary in order to achieve the goal.

4. Utility based agents
A utility maps a state to a real number, so now the agent can actually obtain a measurement of how successful it is being in achieving an objective.

All of these can be turned into learning agents.

5. Learning Agents

- A learning agent in AI is the type of AI 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 adapt automatically through learning.
- A learning agent has mainly four conceptual components, which are:

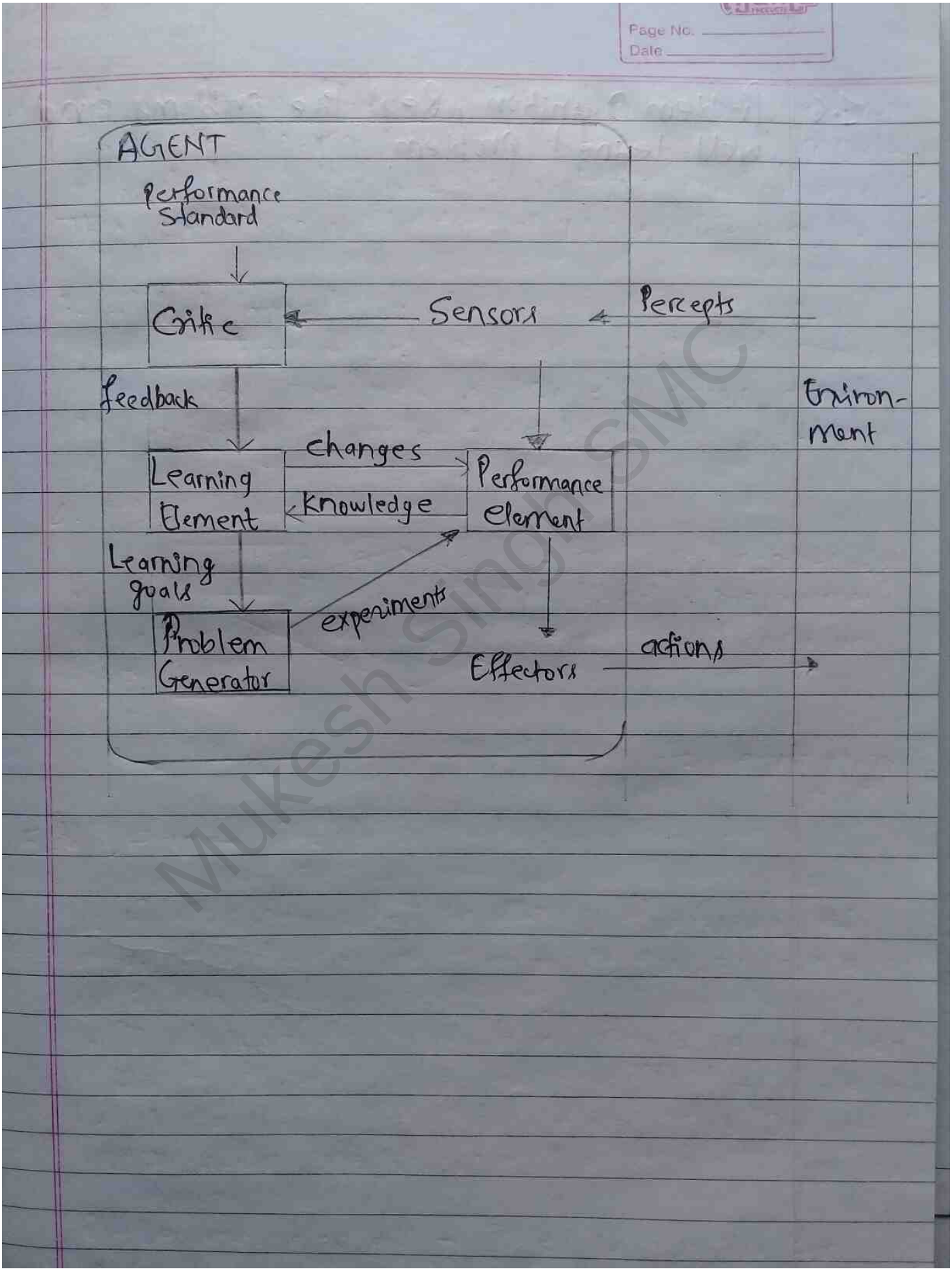
i) Learning element → It is responsible for making improvements by learning from environment.

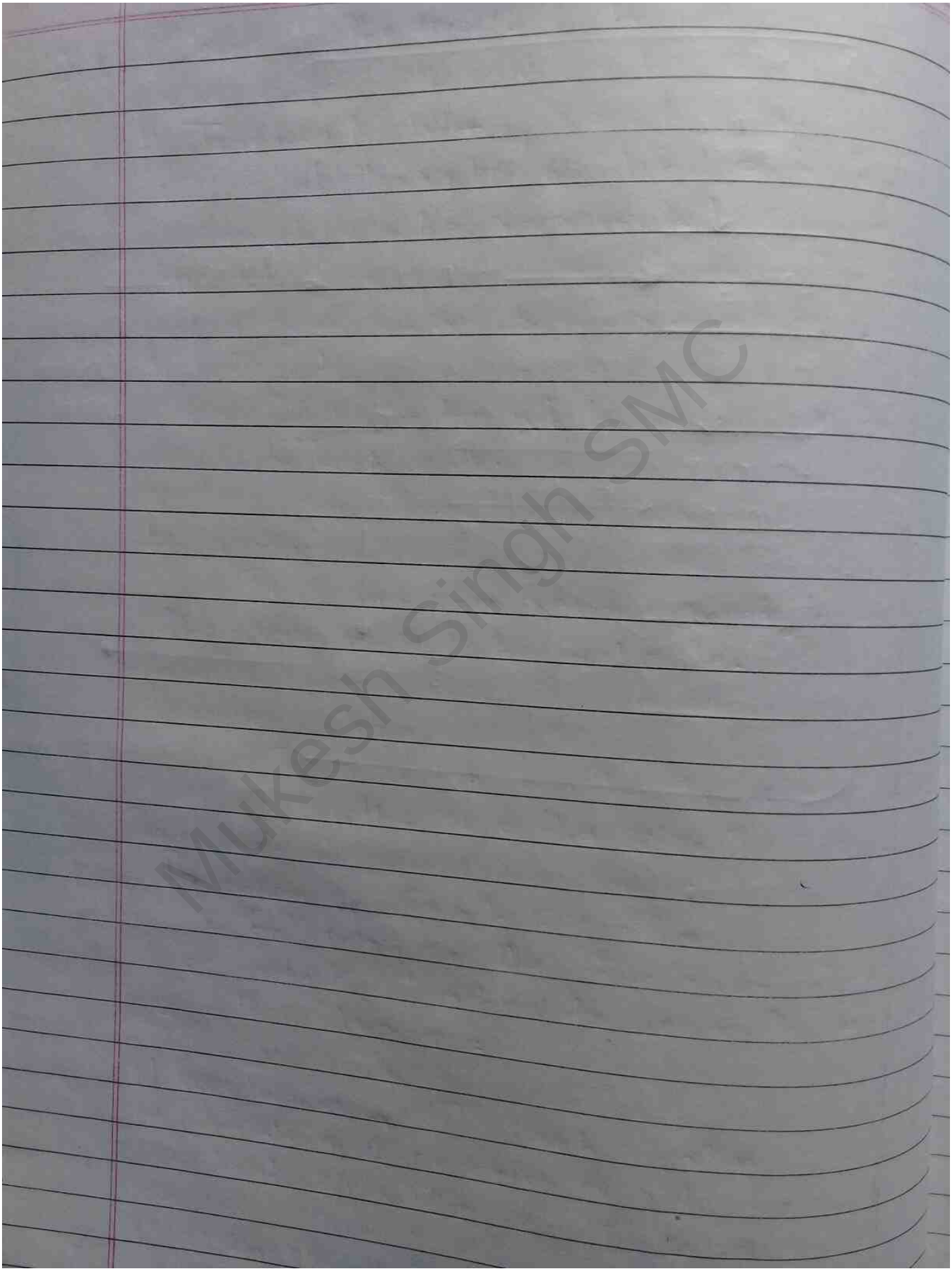
ii) Critic → Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.

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

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

Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.





# Unit-2
## Search Techniques

In Artificial Intelligence, Search techniques are universal problem-solving methods. Rational agents or problem-solving agents in AI mostly used these search techniques or algorithms to solve a specific problem, and provide the best result.

### Search Algorithm Terminologies

#### # Search

Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:

a) Search Space → Search space represents a set of possible solutions, which a system may have.

b) Search State → It is a state from where agent begins the search.

c) Goal test → It is a function which observe the current state and returns whether the goal state is achieved or not.

#### # Search Tree

A tree representation of search problem is called Search Tree. The root of the search tree is the root node which is corresponding to the initial state.

#### # Actions

It gives the description of all the available

actions to the agent.

# Transition Model -
A description of what each ~~do~~ action do, can be represented as a transition model.

# Path Cost
If a function which assigns a numeric cost to each path.

# Solution
It is an action sequence which leads from the start node to the goal node.

# Optimal Solution
If a solution has the lowest cost among all solutions, then it is the optimal solution.

Properties of Search Algorithms
Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

i) Completeness → A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

ii) Optimality → If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution

for is said to be an optimal solution.

iii) Time Complexity → Time Complexity is a measure of time for an algorithm to complete its task.

iv) Space Complexity → It is the maximum storage space required at any point during the search, as the complexity of the algorithm.

Types of Search algorithms

Based on the search problems we can classify the search algorithms into Uniformed (Blind) Search and informed (Heuristic) search algorithms.

## 2.1. Uninformed Search techniques : depth first search, breadth first search, depth limit search, Iterative deepening search

Uninformed search is a class of general-purpose search algorithms which operates in brute-force-way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. There is no information on the goal node other than the one provided in the problem definition.

Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.

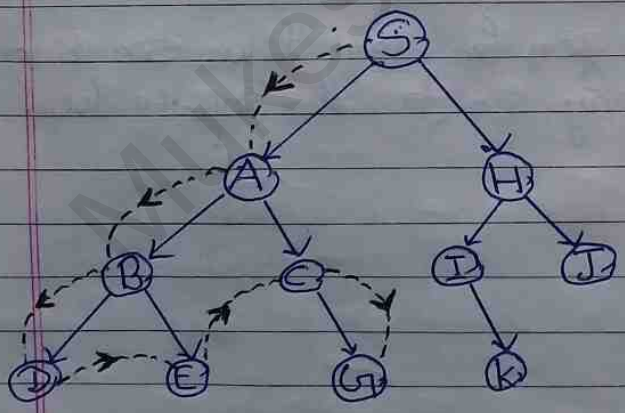
Following are the various types of uninformed search algorithms:

## 1. Depth First Search (DFS)

- DFS is a recursive algorithm for traversing a tree or graph data structure.
- It is based on the concept of LIFO.
- It is called DFS because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

Note - Backtracking is an algorithm technique for finding all possible solutions using recursion.

Example:




In the above search tree, we have shown the flow of depth-first search, and it will follow the order as:

Root node → Left node → right node

It will start searching from node S, and

traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking, it will traverse node C and then G, and here it will terminate as it found goal node.

Path → S --> A --> B --> D --> E --> C --> G

Advantages:

→ DFS requires very less memory.
→ It takes less time to reach the goal node.

Disadvantages:

→ There is the possibility that many states keep re-occuring, and there is no guarantee of finding a solution.

→ DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

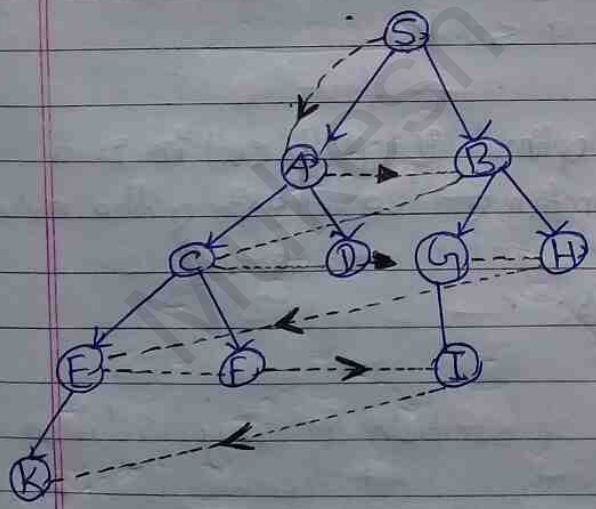
## 2. Breadth First Search (BFS)

→ BFS is the most common search strategy for traversing a tree or graph. This algorithm algorithm searches breadthwise in a tree or graph, so it is called breadth-first-search.

→ It is implemented using queue data structure that works on the concept of FIFO.

→ BFS algorithm ~~starts~~ searching from the root of the tree and expands all successor node at the current level before moving to nodes of next level.

Example:




→ Level 0
→ Level 1
→ Level 2
→ Level 3
→ Level 4

In the above tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path ~~of~~ which is shown by the dotted arrow, and

the traversed path will be:
S --> A -> B -> C -> D -> G --> H -> E -> F -> I -> K.

Time Complexity → Time complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest node. Where the d = depth of shallowest solution and b is a node at every state.
$T(b) = 1 + b^2 + b^3 + \dots + b^d = O(b^d)$

Space Complexity → Space Complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeteleness → BFS is Complete, which means if the shallowest goal node is at some finite depth then BFS will find a solution.

Optimality → BFS is optimal if path cost is a non-decreasing function of the depth of the node.

# Advantages of BFS:

→ BFS will provide a solution if any solution exists.
→ If there are more than one solution for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

Disadvantages of BFS

→ It requires lots of memory since each level of the tree must be saved into memory to expand the next level.

→ BFS needs lots of time if the solution is far away from the root node.

Remaining of Depth First Search :

Completeness → DFS is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity → Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

$$T(n) = 1 + n^2 + n^3 + \ldots + n^m = O(n^m)$$

Where, m = maximum depth of any node and this can be much larger than d (shallowest solution depth).

Space Complexity → DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is O(bm).

Optimal → DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach the goal node.

## 3. Depth limit Search

Depth-limit search algorithm is similar to depth-first-search with a predetermined limit. Depth-limit search can solve the drawback of the infinite path in the DFS.

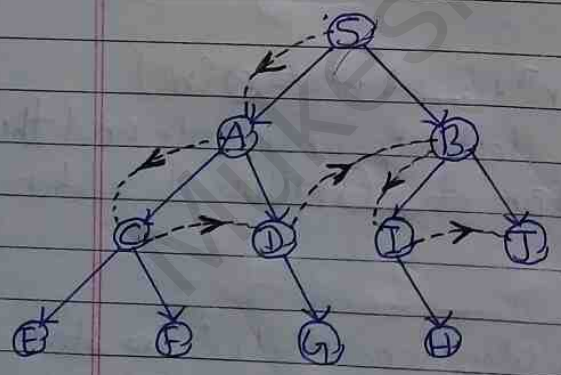
In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

Depth-limit search can be terminated with two conditions of failure:

→ Standard failure value: It indicates that problem does not have any solution.

→ Cutoff failure value: It defines no solution for the problem within a given depth limit.

Example




→ Level 0

→ Level 1

→ Level 2

→ Level 3

Path :- S -> A -> C -> D -> B -> I -> J

Completeness → DLS algorith Complete if the solution is above the depth-limit.

Time Complexity → Time Complexity of DLS algorithm

is ~~O(b×d)~~ $O(b^d)$.

Space Complexity ⇒ Space complexity of DLS algorithm is $O(b \times l)$.

Optimal → Depth-limited search can be viewed as a special case of DFS, and if is also not optimal even if $l > d$.

# Advantages of DLS
→ It is memory efficient.

# Disadvantages of DLS:
→ DLS also have disadvantage of ~~inform~~ incompleteness.

→ It may not be optimal if the problem has more than one solution.

## 4. Iterative Deepening Search

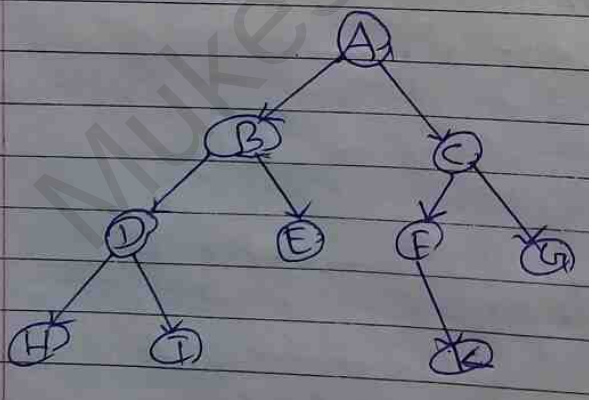
The Iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

This algorithm performs depth-first search up to a certain "depth-limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

The search algorithm combines the benefits of BFS's fast search and DFS's memory efficiency.

The iteration iterative search algorithm is useful uniform search when search space is large, and depth of goal node is unknown.

Example




→ Level 0

→ Level 1

→ Level 2

→ Level 3

1st iteration ---> A
2nd iteration ---> A, B, C
3rd iteration ---> A, B, C, D, E, C, F, G
4th iteration ---> A, B, D, H, I, E, C, F, K, G

In the fourth iteration, the algorithm will find the goal node.

Completeness → This algorithm is complete in which fifth branching is finite.

Time Complexity → Let's suppose b is the branching factor and depth is d then the worst-case time complexity is: $O(b^d)$

Space Complexity → The space complexity of IDDFS will be $O(bd)$.

Optimal → IDDFS algorithm is optimal if path cost is a non-decreasing function of the depth of the node.

#Advantages of
→ It Combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

# Disadvantages:
→ The main drawback of IDDFS is that it repeat ~~the~~ all the work of previous phase.

## 2.2 Heuristic search techniques: Greedy Best first search, A* search, Hill Climbing, Game playing, Adversial search techniques-mini-max procedure, alpha-beta pruning.

The informed search algorithm is ~~called~~ more useful for large search space. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.

Informed search algorithms contains an array of knowledge such as how far we are from the goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the search space and find more ~~efficiently~~ the goal node.

Following are the various types of informed search algorithms:

1. Greedy Best first search
2. A* search
3. Hill Climbing
4. Game playing
5. Adversial search technique-mini-max procedure
6. Alpha-beta pruning

1. Greedy Best First Search

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of DFS and BFS algorithms. It uses the heuristic functions and search. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

$$f(n) = h(n)$$

where,

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

The greedy best first search algorithm is implemented by the priority queue.

Algorithm steps:

Step 1: Place the starting node in the OPEN list.

Step 2: If the OPEN list is empty, Stop and return failure.

Step 3: Remove node n (node with best score) from OPEN list and place it in the CLOSED list.

Step 4: Expand node n and generate the successors of node n.

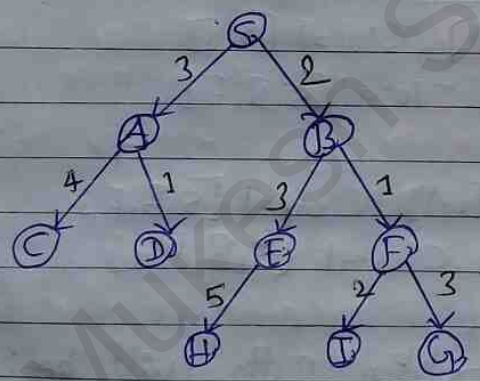
Step 5: If any successor node is goal node, then return success and terminate the search, else proceed to step 6.

Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.

Step 7: Return to step 2.

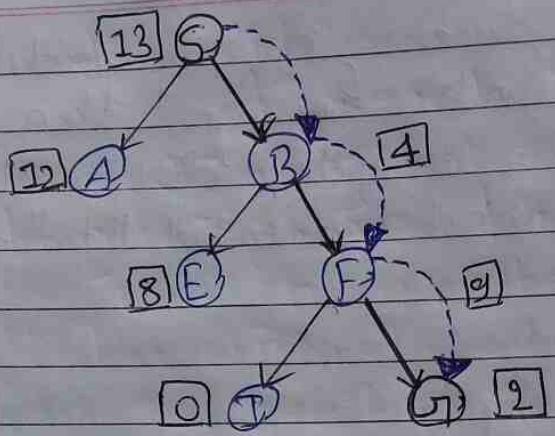
Example

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function $f(n) = h(n)$, which is given in the below table:




| node | H(n) |
|---|---|
| A | 12 |
| B | 4 |
| C | 7 |
| D | 3 |
| E | 8 |
| F | 2 |
| GH | 4 |
| I | 9 |
| S | 13 |
| G | 0 |

In this example, we are using two lists which are OPEN and CLOSED lists. Following are the iteration for traversing the above example.




Expand the nodes of S and put it in the CLOSED list.

Initialization : Open [A, B], Closed [S]

Iteration 1 : Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence, the final solution path will be:
S ---> B ---> F --> G

Time Complexity → The worst case time complexity of Greedy best first search is $O(b^m)$.

Space Complexity → The worst case space complexity of GBFS is $O(b^m)$. Where, m is the maximum depth of the search space.

Page No.
Date

Complete → GBFS is also incomplete, even if the given state space is finite.

Optimal → GBFS is not optimal.

# Advantages of GBFS:

→ GBFS can switch between BFS and DFS by gaining the advantages of both the algorithms.

→ This algorithm is more efficient than BFS and DFS algorithm.

# Disadvantages of GBFS:

→ It can ~~have~~ behave as an unguided depth-first search in the worst case scenario.

→ It can get stuck in a loop as DFS.

→ This algorithm is not optimal.

2. A* Search

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently.

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both cost as following and this sum is called as fitness number.

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

- $f(n)$: Estimated cost of the cheapest solution
- $g(n)$: Cost to reach node n from start state
- $h(n)$: Cost to reach from node n to goal node

Algorithm of A* Search

Step 1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

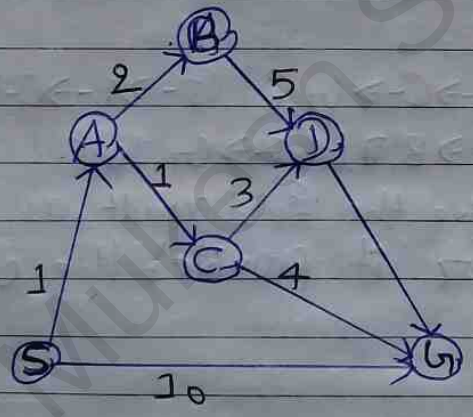
Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise,

Step 4: Expand node n and generate all of its successors, and put ~~it~~ n into the closed list. For each successor 'n', check whether 'n' is already in the OPEN or CLOSED list, if not then compute evaluation function for 'n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

Example:

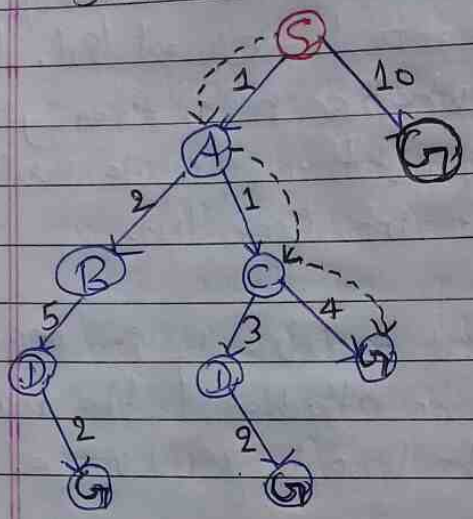



| State | h(n) |
|---|---|
| S | 5 |
| A | 3 |
| B | 4 |
| C | 2 |
| D | 6 |
| G | 0 |

In this ~~trav~~ example, we have traverse graph given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula $f(n) = g(n) + h(n)$, where g(n) is the cost to reach any node from start state.

Here, we will use OPEN and CLOSED list.

Soln:




Initialization : {(S,5)}

Iteration 1: {(S->A,4),(S->G1,10)}

Iteration 2: {(S->A->C,4),(S->A->B,7),(S->G1,10)}

Iteration 3: {(S->A->C->G1,6),(S->A->C->D,11)
(S->A->B,7),(S->G1,10)}

Iteration 4 will give the final result, as S--->A--->C--->G1 it provides the optimal path with cost 6.

Complete → A* search algorithm is complete as long as:
- Branching factor is ~~line~~ finite.
- Cost at every action is fixed.

Optimal → A* search algorithm is optimal if it follows below two conditions:
- Admissible: the first condition requires for optimality is that h(n) should be an admissible

heuristic for A* tree search. An admissible heuristic is optimal in nature.

- Consistency → Second required condition is consistency for only A* graph-search.

Time Complexity → The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d) where b is the branching factor.

Space Complexity → The space complexity of A* search algorithm is O(b^d).

# Advantages of A* search algorithm

→ It is optimal and Complete.
→ This algorithm can solve very complex problems.

# Disadvantages of A* search algorithm:

→ A* search algorith has some security issues.

→ It is not practical for various large-scale problems.

→ Memory requirement.

## 3. Hill Climbing

- Hill climbing is algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a high value.

- It is also called greedy local search as it only looks to its good immediate neighbour state and not beyond that.

- Hill climbing is mostly used when a good heuristic is available.

### Features of Hill Climbing:

i) Generate and Test Variant → Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.

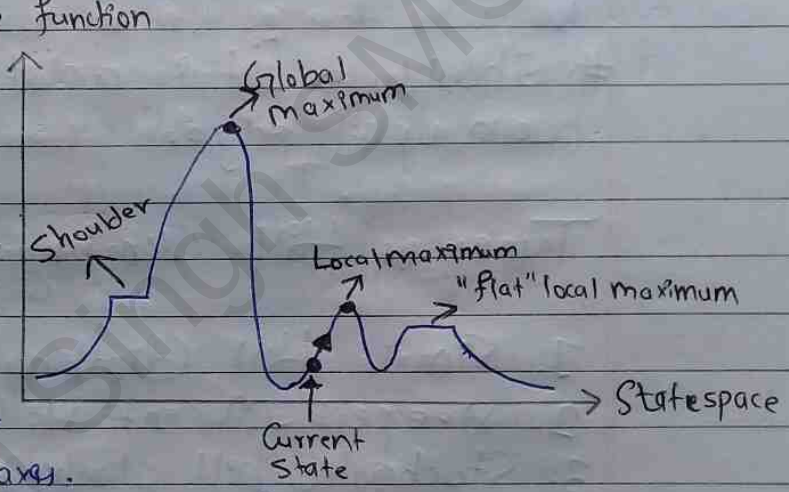
ii) Greedy Approach → Hill-climbing ~~is the~~ algorithm search moves in the direction which optimizes the cost.

iii) No backtracking → It does not backtrack the search space, as it does not remember the

previous states.

## State-Space Diagram for Hill Climbing

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.




On Y-axis we have taken the function which can be an Objective function or Cost function, and state-space on the X-axis.

If the function on Y-axis cost then, the goal of search is to find the global minimum and local ~~maximum~~ minimum.

If the function of Y-axis is Objective function, then the ~~global~~ goal of the search is to find the global maximum and local maximum.

### Different regions in the state space diagram:

# Local Maximum → It is a state which is better than its neighbour states, but there is also another state which is higher than it.

# Global ~~minimum~~ Maximum → It is the best possible state of state space ~~diagram~~ landscape. It has

the highest value of objective function.

# Current State → It is a state in a landscape diagram where an agent is currently present.

# Flat local maximum → It is a flat space in the landscape where all the neighbour states of current states have the same value.

# Shoulder → It is a plateau region that has an uphil edge.

Types of Hill Climbing:

1. Simple Hill Climbing
2. Steepest-Ascent Hill-Climbing
3. Stochastic hill climbing

## 4. Game Playing

Game playing is the design of AI to be able to play more than one game successfully. Game playing is important domain of AI. Games don't require much knowledge; the only knowledge we need to be provided 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 moves 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 procedure that improve -

- General procedure: So that only good moves are generated.
- Test procedure: So that the best move can be explored first.

The most common search technique in game playing is Minimax Search procedure / algorithm. It is depth-first depth-limited search procedure. It is used for games like Chess and tic-tac-toe.

Minimax algorithm uses two functions:

MOVEGEN → It generates all the possible moves that can be generated from the current position.

STATICEVALUATION → It returns a value depending upon the goodness from the viewpoint of two-player.

# Working of minimax algorithm:

This algorithm is a two player game, so we call the first player as PLAYER1 and second player as PLAYER2. The value of each node is backed-up from its children. For PLAYER1, the backed up value is the maximum value of its children and for PLAYER2, the backed up value is the minimum value of its children. It provides most promising move to PLAYER1, assuming that the ~~Player~~ PLAYER2 has make the best move. It is a recursive algorithm, as same procedure occur at each level.

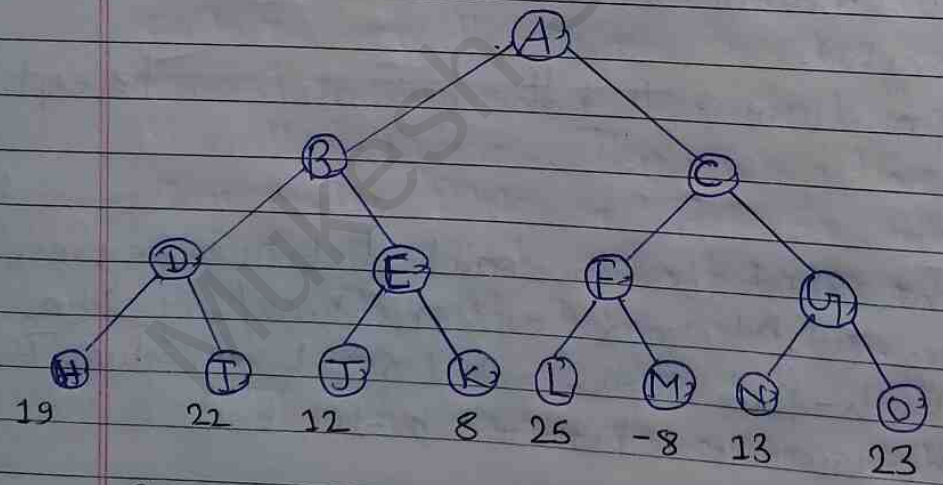



Figure 1: Before backing up of values

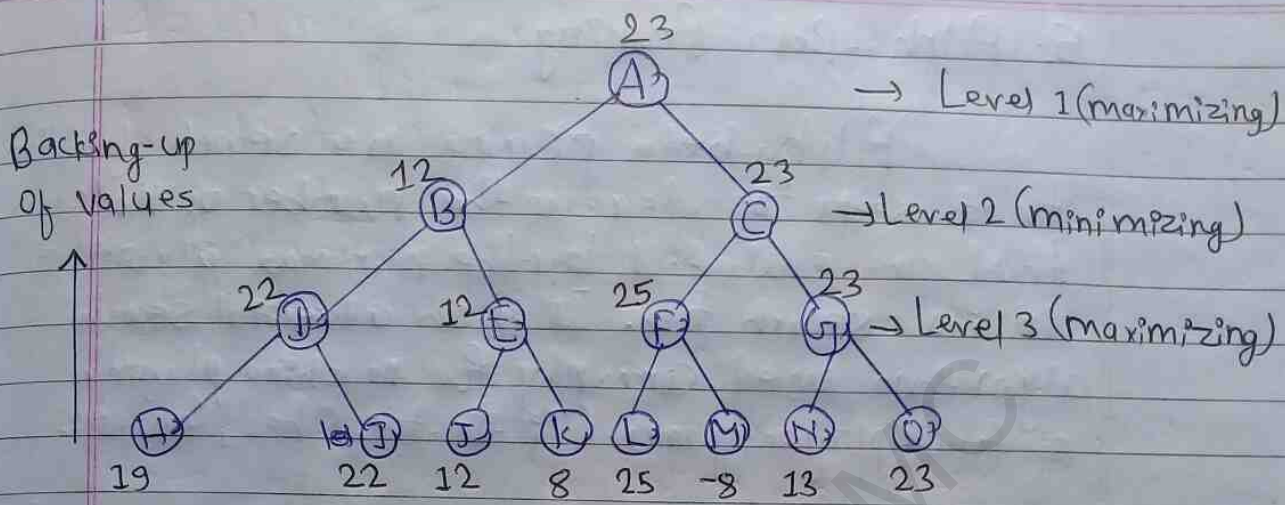



Fig 2: After backing up of values

We assume that PLAYER 1 will start the game. 4 levels are generated. The value of nodes H, I, J, K, L, M, N, O is provided by STATICEVALUATION function. Level 3 is maximizing level, so all nodes of level 3 will take maximum values of their children. Level 2 is minimizing level, so all nodes will take minim values of their children. This process continues. The value of A is 23. That means A should choose C more to win.

Limitation of minimax algorithm

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from alphabeta pruning.

## 5. Alpha-Beta Pruning

Alpha-beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc).

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

There is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and Beta for future expansion, so it is called alpha-beta pruning. It is also called alpha-beta algorithm.

The two parameter can be defined as:

a.) Alpha → The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is $-\infty$.

b) Beta → The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is $+\infty$.

- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes, which are not really affecting the final decision but making algorithm slow. Hence, by pruning these nodes, it makes the algorithm fast.

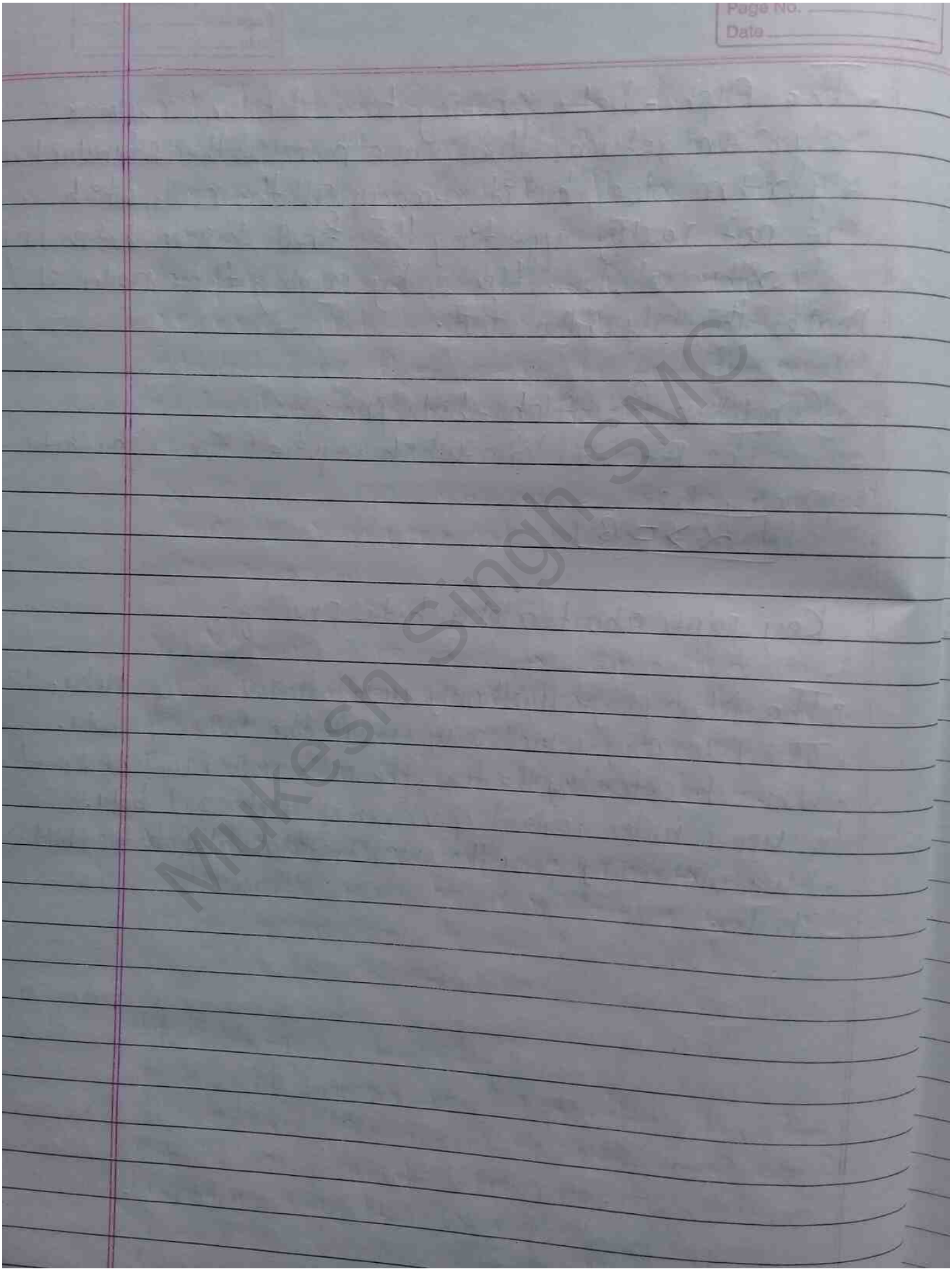
Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

$$\alpha >= \beta$$

Key points about alpha-beta pruning:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta value to the child nodes.


Page No.
Date
Mukesh Singh SMC

# Unit-3

## Knowledge, Reasoning and Planning

Page No.
Date

### #Knowledge

Knowledge is awareness or familiarity gained by experiences of facts, data and situations.

Following are the types of knowledge in AI:

i) Declarative knowledge → to know about something.

ii) Procedural knowledge → is responsible for knowing how to do something.

iii) Meta-knowledge → it is the knowledge about other types ~~types~~ of knowledge.

iv) Heuristic knowledge → representing knowledge of some experts in a field or subject.

v) Structural knowledge → is basic knowledge to problem-solving.

### #Reasoning

Reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs.

OR, It is a way to infer facts from existing data. It is general process of thinking rationally, to find valid conclusions.

Following are the types of reasoning in AI:

i) Deductive reasoning → is deducing new information from logically related known information.

ii) Inductive Reasoning → is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization.

iii) Abductive reasoning → is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.

iv) Common Sence reasoning → is an informal form of reasoning which can be gained through experiences.

v) Monotonic reasoning → Once the conclusion is taken then it will remain the same even if we add some other information to existing information in our knowledge base.

vi) Non-monotonic reasoning → Some conclusions may be invalidated if we add some more information to our knowledge base.

# Planning

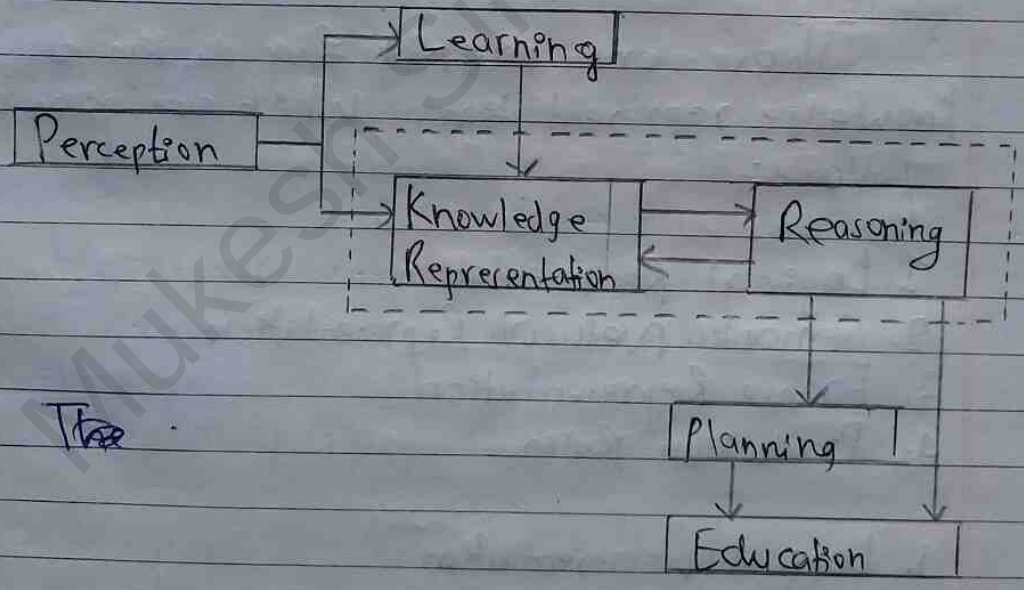
Planning in AI is about the decision making tasks performed by the robots or computer programs to achieve a specific goal.

# Knowledge Representation

Knowledge representation and reasoning (KR, KRR) is the field of AI dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language.

## AI Knowledge Cycle:

An artificial intelligence system has the following components for displaying intelligent behaviour as shown in the figure:




The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory ~~information~~ input.

The Learning Component Which is responsible for learning from data captured by perception component. In the complete cycle the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of knowledge representation and reasoning.

Techniques of Knowledge Representation:

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules

1. Logical Representation

Logical representation is a language with definite rules which deals with propositions and has no ~~definite~~ ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics

which supports the sound inference. Each sentence can be translated into logics by using syntax and semantics.

Syntax:
- Syntax are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

Semantics
- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.

Logical representation can be categorized into mainly two categories:
a. Predicate Logic
b. Propositional logic.

Advantages of Logical Representation:
1. It enables us to do logical reasoning.
2. It is the basis for programming languages.

Disadvantages of Logical Representation:
1. They have some restrictions and are challenging to work with.
2. This technique may not be very natural, and inference may

not be so efficient.

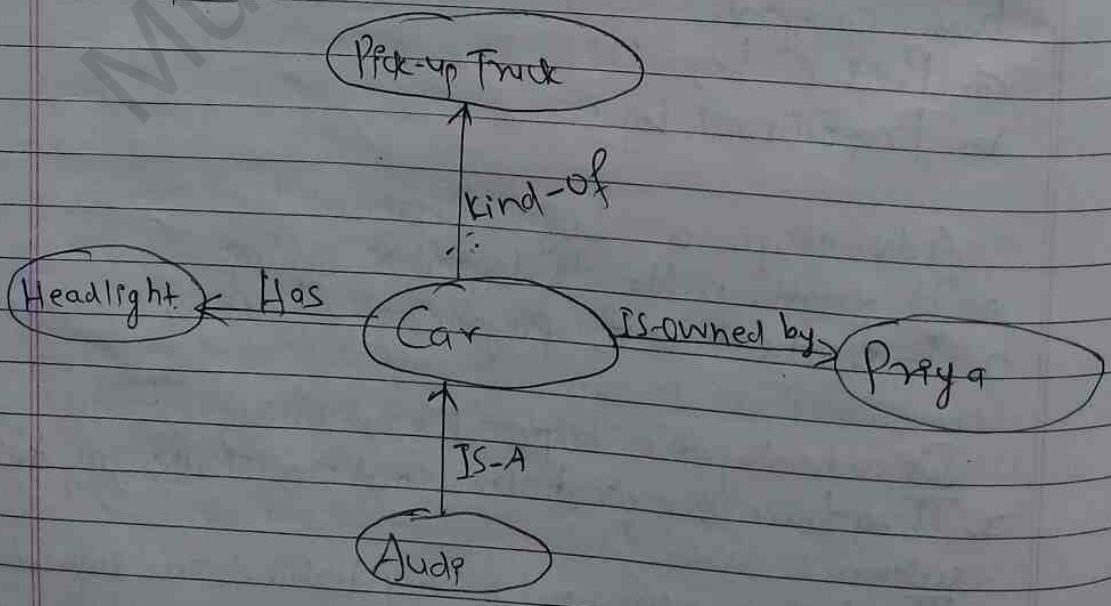
2. Semantic Network Representation

Semantic networks are the alternative of predicate logic for knowledge representation. In semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks ~~may contain~~ are easy to understand and can be easily extended.

This representation consists of mainly two types of relations:

a. Is-A relation (Inheritance)
b. kind of relation

Example



Advantages
→ Easy to understand and visualise.
→ Are natural representation of knowledge.
→ It conveys meaning in a transparent manner.

Disadvantages
→ It takes more computational time at runtime.
→ There networks are not intelligent and depend on the creator of the system.

3. Frame Representation

A frame is a record like structure which consists of a collection of attribute and its values to describe any entity in the world. Frames are the AI data structures which divides knowledge into substructures by representing stereotypes situations. It consists of collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facts. A frame is also known as slot-filter knowledge representation in AI.

Example
Lets take an example of a frame for a student:

| Slots | Filter |
|---|---|
| Name | Mukesh |
| Profession | student |
| Age | 25 |
| Marital status | Single |
| Weight | 60 |

Example 3

Let's take an example of a frame for a book:

| Slots | Filters |
|---|---|
| Title | Artificial Intelligence |
| Genre | Computer Science |
| Author | Peter Norving |
| Edition | Third Edition |
| Year | 1996 |
| Page | 1152. |

Advantages of frame representation

→ It makes programming easier by grouping the related data.
→ It is easy to understand and visualize.
→ It is easy to add slots for new attributes and relations.

Desadvantages

→ The mechanism cannot be easily procersed.
→ It has much generalized approach.

4. Production Rules

Production rule system ~~consists~~ has of three main parts:

- The set of production rules
- Working memory
- The recognize-act-cycle

Production rules system consists of (condition, action) pairs which mean, "If condition then action".

In production rules, agent checks for the action and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. Whereas, the action part carries out the associated problem-solving steps. This complete process is called a recognize-cycle.

Example

→ IF (at bus stop AND bus arrives) THEN action (get into the bus).
→ IF (on the bus AND paid AND Empty seat) THEN action (sit down).
→ IF (on bus AND unpaid) THEN action (pay charges).
→ IF (bus arrives at destination) THEN action (get down from the bus).

Advantages of Production rule:

→ The production rules are expressed in natural language.
→ They are highly modular, so we can easily add, remove or modify an individual rules.

Disadvantages

→ It does not store the result of the problem for the future uses.
→ During the execution of the program, many rules may be active. Thus, rule-based production system are inefficient.

3.1. Formal logic connectives : truth tables, syntax, semantics, tautology, validity, well-formed formula.

A logical connective is a symbol or word used to connect two or more sentences in a grammatically valid way, such that the value of the compound sentence produced depends only on that of the original sentences and on the meaning of the connective.

It connects two ~~individual~~ Undividable Simple sentences or expresses a sentence in a logical sense. Complex sentences can be created using logical connectives. There are 5 types of Connectives given below:

1. Negation → It represents a negative condition. P is a positive statement and ¬P indicates NOT condition.
Example: Today is Monday (P).
Today is not a Monday (¬P).

2. Conjunction → It joins two statements P, Q with AND clause. It is denoted by : (P ∧ Q).
Example
Ram is intelligent (P).
Ram is hardworking (Q).
Ram is intelligent and hardworking (P ∧ Q) is represented by (P ∧ Q).

3. Disjunction → It joins two statements P, Q with OR clause. It is denoted by (P v Q).

Example:

Rock is a wrestler (P).

Rock is an actor (Q).

Rock is a wrestler or an actor is represented by (P v Q).

4. Implication → Sentence (Q) is dependent on sentence (P), and it is called implication, denoted by P → Q. It follows the rule of then clause. If sentence P is true, then sentence Q is true. The condition is unidirectional.

Example

If it is Sunday (P) then I will go to College (Q), and it is represented as P → Q.

5. Bi-Conditional → A sentence such as P <=> is a biconditional sentence. If a conditional sentence and its converse are true, then it is called ~~bidirect~~ biconditional connective. (P → Q and Q → P i.e. P <=> Q).

Example

~~If I am breathing, then I am alive.~~

I am breathing (P).

I am alive (Q).

~~It can be represent~~

If I am breathing, then I am alive and it can be represented as: P <=> Q.

## Truth Tables:

We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called Truth Table. Following are the truth tables for all logical connectives:

For Negation (¬ or ~)

| P | ¬P |
|---|---|
| True | False |
| False | True |

For Conjunction (∧)

| P | Q | P ∧ Q |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

For disjunction (∨)

| P | Q | P ∨ Q |
|---|---|---|
| True | True | True |
| False | True | True |
| True | False | True |
| False | False | False |

For Implication (→)

| P | Q | P→Q |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | True |
| False | False | True |

For Biconditional (⇔)

| P | Q | P⇔Q |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | True |

# Syntax
- Rules for constructing legal sentences in the logic.
- Which symbols we can use (English: letters, punctuation)
- How we are allowed to combine symbols.

# Semantics
- How we interpret (read) sentences in the logic.
- Assigns a meaning to each sentence.

Example:
"All lecturers are seven foot tall"
- A valid sentence (syntax).
- And we can understand the meaning (semantics).
- This sentence happens to be false (there is a counter example).

In logic Connectives:

- Syntax
  - Propositions, e.g. "it is raining".
  - Connectives: and, or, not, implies, iff (equivalent)
    $\wedge$, $\vee$, $\neg$, $\rightarrow$ $\leftrightarrow$
  - Brackets, T(true) and F(False)

- Semantics (Classical i.e. Boolean)
  - Define how connectives affect truth.
    - "P and Q" is true if and only if P is true and Q is true.
  - Use truth tables to work out the truth of sentences (statements).

# Tautology

A tautology is a formula which is "always true"..that is, it is true for every assignment of truth values to its simple components. You can think of a tautology as a rule of logic.

The opposite of a tautology is a contradiction, a formula which is "always false". In other words, Contradiction is false for every assignment of truth values to its simple components.

Example: Show that $(P \to Q) \vee (Q \to P)$ is a tautology.

Construct the truth table for: $(P \to Q) \vee (Q \to P)$

| P | Q | $P \to Q$ | $Q \to P$ | $(P \to Q) \vee (Q \to P)$ |
|---|---|---|---|---|
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | T | F | T |
| F | F | T | T | T |

The last column contains only T's. Therefore, $(P \to Q) \vee (Q \to P)$ is a tautology.

# Validity

Validity, in logic, is the property of an argument consisting in the fact that the truth of the premises logically guarantees the truth of the conclusion.

Whenever the premises are true, the conclusion must be true, because of the form of the argument.

True premises and a valid argument guarantee a true conclusion.

# Well formed formula

Well-formed formula (WFF) is a finite sequence of symbols from a given alphabet that is part of a formal language. A formal language can be identified with the set of formulas in the language.

WFF can also be defined as a valid string with statement letters, connectives and parentheses.

The term WFF is defined as follows:

~~1. Any capital is a WFF.~~
~~2. A dash (~) followed by a WFF is a WFF.~~
~~3.~~

1. Any statement variable is a WFF.
2. For any WFF $\alpha$, $\sim\alpha$ is a WFF.
3. If $\alpha$ and $\beta$ are WFFs then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ and $(\alpha \leftrightarrow \beta)$ are WFFs.
4. A finite string of symbols is a WFF only when it is constructed by steps 1, 2, and 3.

Example of Well formed formula:

- By definition of WFF.
  - WFF: $\neg(P \wedge Q)$, $(P \rightarrow (P \vee Q))$, $(\neg P \wedge Q)$, $((P \rightarrow Q) \wedge (Q \rightarrow R)) \leftrightarrow (P \rightarrow R))$, etc.
  - ~~not WFF.~~
  - Not WFF:
    1. $(P \rightarrow Q) \rightarrow (\wedge Q)$: $(\wedge Q)$ is not a WFF.
    2. ~~(P Q) :b~~ $(P \rightarrow Q$: but $(P \rightarrow Q)$ is a WFF etc.

3.2 Propositional Logic, Inference with PL: Resolution, Backward Chaining and Forward Chaining.

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

Examples.
a) It is Sunday.
b) The sun rises from West (False proposition).
c) 3+3=7 (False proposition).
d) 5 is a prime number.

Following are some basic facts about Propositional logic:

→ Propositional logic is also called Boolean logic as it works on 0 and 1.
→ Propositions can be either true or false but not both.
→ Propositional logic consists of an object, relation, or function, and logical connectives.
→ These connectives are also called logical operators.
→ The connectives and propositions are the basic elements of the propositional logic.
→ Connectives can be said as logical operators which connects two sentences.
→ A proposition formula which is always true is called tautology, and it is also called valid sentence.

→ A proposition which is always false is called contradiction.

→ Statements which are questions, commands or opinions are not propositions such as "Where is Rohini", "How are you", "What is your name", are not propositions.

## Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of propositions:

a) Atomic Propositions

Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example

i) 2+2 is 4., it is an atomic proposition as it is a true fact.

ii) "The sun is cold" is also a proposition as it is a false fact.

b) Compound Propositions

Compound propositions are constructed by combining simple or atomic propositions, using parenthesis and logical connectives.

Example

i) "It is raining today, and street is wet."

ii) "Kishor is a student, and he studies ICT".

Logical Connectives:

Logical connectives are used to connect two simpler propositions logically.

Following is the summarized table for Propositional logic Connectives:

| Connective Symbol | Word | Technical term | Example |
|---|---|---|---|
| $\wedge$ | AND | Conjunction | $A \wedge B$ |
| $\vee$ | OR | Disjunction | $A \vee B$ |
| $\neg$ or $\sim$ | NOT | Negation | $\neg A$ or $\neg B$ |
| $\rightarrow$ | Implies | Implication | $A \rightarrow B$ |
| $\Longleftrightarrow$ | if and only if | Biconditional | $A \Longleftrightarrow B$ |

# Inference with Propositional Logic:

Generating the conclusions from evidence and facts is termed as Inference.

Inference rules:

Inference rules are the templates for generating valid arguments. Rules of inference provide the templates or guidelines for constructing valid arguments from the statements that we already have.

Types of Inference rules:

1. Modus Ponens:

It states that if P and $P \rightarrow Q$ is true, then we can infer that Q will be true. It can be represented as:

$$\frac{P \quad P \rightarrow Q}{\therefore Q}$$

2. Modus Tollens:
It states that if $P \to Q$ is true and $\neg Q$ is true, then $\neg P$ will also be true. It can be represented as:

$$\frac{P \to Q \quad \neg Q}{\therefore \neg P}$$

3. Hypothetical Syllogism:
It states that if $P \to R$ is true whenever $P \to Q$ is true, and $Q \to R$ is true. It can be represented as:

$$\frac{P \to Q \quad Q \to R}{\therefore P \to R}$$

4. Disjunctive Syllogism
It states that if $P \vee Q$ is true and $\neg P$ is true, then $Q$ will be true. It can be represented as:

$$\frac{P \vee Q \quad \neg P}{\therefore Q}$$

5. Addition
It states that if $P$ is true, then $P \vee Q$ will be true.

$$\frac{P}{\therefore P \vee Q}$$

6. Simplification

It states that if $P \wedge Q$ is true, then Q or P will also be true. It can be represented as:

$$\frac{P \wedge Q}{\therefore P} \quad \text{or,} \quad \frac{P \wedge Q}{\therefore Q}$$

7. Resolution

It states that if $P \vee Q$ and $\neg P \wedge R$ is true, then $Q \vee R$ will be also be true. It can be represented as:

$$\frac{P \vee Q \\ \neg P \wedge R}{\therefore Q \vee R}$$

Inference

Generating the ~~evidence from~~ conclusions from the evidence and facts is termed as Inference.

# Forward Chaining and Backward Chaining:

In AI, forward and backward chaining is one of the topics. But before understanding forward and backward chaining lets first understand from where these two terms came.

## Inference Engine:

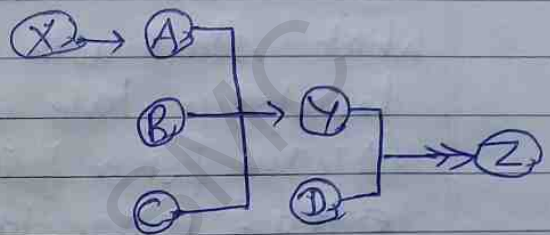
The inference engine is the component of the intelligent/expert system in AI, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was the part of the expert system.

- The inference engine compares each rule stored in the knowledge base with facts contained in the database.
- When the IF (condition) part of the rule matches a fact, the rule is fired and its THEN (action) part is executed.
- The matching of the rules IF parts to the facts produces inference chains.
- The inference chain indicates how an expert system applies the rules to reach a conclusion.

Example of inference chain:

Rule 1: IF X is true
THEN A is true

Rule 2: IF A is true
AND B is true
AND C is true
THEN Y is true




Rule 3: IF Y is true
AND D is true
THEN Z is true

Inference engine commonly proceeds in two modes, which are:

a. Forward Chaining
b. Backward Chaining

## A. Forward Chaining

Forward chaining is also known as forward deduction or forward reasoning method when using an inference engine.

Forward Chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their Conclusion to the known facts. This process repeats until the problem is solved.

Properties

- It is a bottom-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward Chaining approach is also called as data-driven as the data determines which rules are selected and used.

Page No.
Date

Example:

A He exercises regularly.
A → B If he is exercising, he is fit.
B He is fit

Example 2

Rule 1: IF student is excellent in academics
AND he is good in sports
THEN he is all rounder (Goal).

Rule 2: IF student get marks > 80
AND he is good in general knowledge
THEN he is excellent in academics.

Rule 3: IF student is making centuries
THEN he is good in sports

Given following facts (data):

1. Student gets marks > 80
2. Student is making centuries.
3. He is good in general knowledge.

## B. Backward Chaining

- Backward Chaining is also known as backward deduction or backward reasoning method when using an inference engine. A
- A backward chaining algorithm is a form of reasoning, which starts with the goal, and works backward, chaining through rules to find known facts that support the goal.
- It is a goal driven method of deriving a particular goal from a given knowledge base and set of inference rules.
- The inference system knows the final decision or goal, this system starts from the goal and works backwards to determine what facts must be asserted so that the goal can be achieved.

- Example

B He is fit
A -> B If he is exercising, he is fit.
A He exercises regularly.

Properties of Backward Chaining:

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- The goal is broken into sub-goal or

sub-goals to prove the facts true.

- It is called a goal-driven approach, as a list of goals decides what rules are selected and used.

Example

This is opposite to forward chaining. Here the interpreter tries to match the "THEN" condition instead of IF condition as in the forward chaining.

- Rule 1: IF student is excellent in academics.
  AND he is good in sports.
  THEN he is all rounder (Goal).

- Rule 2: IF student get mark > 80
  AND he is good in general knowledge
  THEN he is excellent in academics.

- Rule 3: IF student is making centuries.
  THEN he is good in sports.

Given facts

1. Student gets mark > 80.
2. Student is making centuries.
3. He is good in general knowledge.

This is forward chaining. To perform backward chaining we take first goal and then find/match facts.

1) Take the ~~goal~~ part from Rule 1:
he is all rounder (Goal) ~~and~~ the
The matching facts are:
- Student is excellent in academics.
- Student is good in sports.

2) Take the goal part from rule 2:
he is excellent in academics (goal)
The matching facts are:
→ Student gets marks > 80.
→ he is good in general knowledge.

3) Take the goal part from rule 3:
he is good in sports (goal)
The matching facts are in rule 3:
→ ~~The~~ student is making centuries.

| Forward Chaining | Backward Chaining |
|---|---|
| 1. It starts from known facts and applies inference rules to extract more data until it reaches to the goal. | 1. It starts from the goal and works backward through inference rules to find the required facts. |
| 2. It is a bottom-up approach. | 2. It is a top-down approach. |
| 3. It is known as data-driven inference technique. | 3. It is known as goal-driven technique. |
| 4. It applies BFS strategy. | 4. It applies DFS strategy. |
| 5. Slow as it has to use all the rules. | 5 Fast as it has to use only a few rules. |
| 6. Can generate an infinite number of possible conclusions. | 6. Generates a finite number of possible conclusions. |
| 7. It operates in forward direction. | 7. It operates in backward direction. |
| 8. It is suitable for planning, monitoring, control and interpretation application. | 8 It is suitable for diagnostic, prescription and debugging application. |

3.3 First Order Predicate Logic (FOPL); quantification, inference with FOPL; by converting into PL (Existential and universal instantiation), Directly with FOPL (unification and lifting resolution, Backward Chaining, Forward Chaining).

## First Order Predicate Logic (FOL)

→ First Order Predicate Logic is another way of knowledge representation in AI. It is an extension to propositional logic.

→ FOPL is sufficiently expressive to represent the natural language statements in a concise way.
→ FOPL is also known as Predicate Logic or First order Logic.
→ FOPL is a powerful language that develops information about the objects in a more easy way and can also expres the relationship between those objects.
→ FOPL (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
objects: A, B, people, numbers, colors, wars, theories, squares, pits etc.
Relations: It can be unary relation such as red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color etc.

Function: Father of, best friend, third wing of, end of etc.

→ As a natural language, FOPL also has two main parts:

a. Syntax
b. Semantics

Syntax of FOPL:

Following are the basic elements of FOPL syntax:

Constant → 1, 2, A, John, Mumbai, Cat ---

Variables → x, y, z, a, b, ...

Predicates → Brother, father, >, ...

Function → sqrt, LeftLegof, ...

Connectives → ∧, ∨, ¬, →, ⇔

Equality → ==

Quantifier → ∀, ∃

Atomic Sentences:
We represent atomic sentences as
Predicate(term 1, term 2, ..., term n).

Example:
Ravi and Ajay are brothers: ⇒ Brothers (Ravi, Ajay).
Chinky is a cat. : ⇒ Cat (Chinky).

Complex Sentences:
Complex sentences are made by combining atomic sentences using connectives.
FOPL logic statements can be ~~defined as~~ a divided into two parts:
- Subject
- Predicate

E.g:

X is an integer
↓
Subject Predicate

Quantifiers in FOPL:
There are two types of quantifiers:

a. Universal Quantifier (∀), (for all, everyone, everything)
→ Universal quantifier is a symbol for logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
It is represented by a symbol, ∀, which resembles an inverted A.

Note: In universal quantifier, we use Implication "→".

Example ①

Every man respects his parent.
predicate is "respect (x, y)"
where, x = man and y = parent
$\forall x$ man(x) → respects (x, parent)

Example ②
All man drink Coffee.
predicate is: "drink (x, y)"
where x = man and y = Coffee
$\forall x$ man(x) → drink (x, Coffee)

b. Existential Quantifier (∃), (for some, at least one)
→ Existential quantifiers are the type of quantifiers, which expres the statement within its scope is true for at least one instance of something.
It is denoted by a logical operator ∃.
Note: In existential quantifier we always use AND or Conjunction symbol (∧).

Example ①
Some boys are intelligent
~~Predicate is:~~
$\exists x$: boys(x) ∧ intelligent (x)

Example ② Some boys play cricket.
predicate is "play (x, y)"
where, x = boys, and y = cricket
$\exists x$ (boys(x) ∧ play (x, cricket).

Some examples of FOPL using quantifier:

1. All birds fly.
   predicate is: "fly (bird)"
   $\forall x \; bird(x) \rightarrow fly(x)$

2. Not all students like both maths and Science.
   predicate is "like(x,y)", where x = student and y = ~~stud~~ subject
   Since there are not all students, so we use $\forall$ with negation, so it is represented as:
   $\neg \forall (x) [student(x) \rightarrow like(x, Maths) \wedge like(x, Science)]$

3. Only one student failed in Maths.
   predicate is "failed (x,y)," where x = student and y = subject.
   Since there is only one student who failed in Maths, so we will use following representation:
   $\exists (x) [student(x) \rightarrow failed(x, Maths) \wedge \forall (y) [\neg (x == y) \wedge student(y) \rightarrow \neg failed(x, Maths)]$.

# Inference in FOPL

Inference in FOPL is used to deduce new facts or sentences from existing sentences. Before understanding the FOPL inference rule, let's understand some basic terminologies used in FOPL:

## Substitution

Substitution is a fundamental operation performed on terms and formulas. The substitution is complex in the presence of quantifiers in FOPL. If we write F[a/x], so it refers to substitute a constant "a" in place of variable "x".

Note → FOPL is capable of expressing facts about some or all objects in the universe.

## Equality

We can use equality symbols which specify that two terms refer to the same object.

Example: Brother(John) = Smith

The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example: $\neg(x=y)$ which is equivalent to $x \neq y$.

FOPL inference rules for quantifier:

As propositional logic we also have inference rules in FOPL, so following are some basic inference rules in FOPL:

- Universal Generalization
- Universal Instantiation
- Existential Instantiation
- Existential introduction

1. Existential Instantiation

→ Existential instantiation is also called as Existential Elimination, which is a valid inference rule in FOPL.

↳ It can be applied only once to replace the existential sentence.

→ The new KB ~~will~~ is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable.

→ This rule states that one can infer $P(c)$ from the formula given in the form of $\exists x\, P(x)$ for a new constant symbol c.

→ The restriction with this rule is that c used in the rule must be a new term for which $P(c)$ is true.

→ It can be represented as: $\frac{\exists x\, P(x)}{P(c)}$

Example

From the given sentence: $\exists x \, Crown(x) \wedge OnHead(x, John)$,

So we can infer: $Crown(K) \wedge OnHead(K, John)$, as long as K does not appear in the knowledge base.

→ The above K is a constant symbol, which is called Skolem Constant.

→ The Existential instantiation is a special case of Skolemization process.

2. Universal Instantiation

→ Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

→ The new KB is logically equivalent to the previous KB.

→ As per UI, we can infer any sentence obtained by substituting a ground term for the variable.

→ The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from $\forall x \, P(x)$ for any object in the universe of discourse.

→ It can be represented as: $\frac{\forall x \, P(x)}{P(c)}$

Example 1

If "Every person like Ice-Cream" $\Rightarrow \forall x \, P(x)$

So we can infer that: "John likes ice-cream" $\Rightarrow P(c)$.

Example 2:

Let is take a famous example, "All kings who are greedy are Evil". So let our knowledge base contains this details as the in the form of FOL:

$\forall x \; King(x) \wedge greedy(x) \rightarrow Evil(x)$.

So from this information, we can infer any of the following statements using Universal Instantiation:

- $King(John) \wedge Greedy(John) \rightarrow Evil(John)$.
- $King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$.
- $King(Father(John)) \wedge Greedy(Father(John)) \rightarrow Evil(Father(John))$.

# Unification

→ Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.

→ It takes two ~~different~~ literals as input and makes them identical using substitution.

→ Let $\psi_1$ and $\psi_2$ be two ~~literals~~ atomic sentences and $\sigma$ be a unifier such that, $\psi_1\sigma = \psi_2\sigma$, then it can be expressed as UNIFY($\psi_1, \psi_2$)

Example : find the MGU for Unify {King(x), King(John)}

Let $\psi_1$ = {King(x)}; $\psi_2$ = King(John)

Substitution $\theta$ = {John/x} is a unifier for these atoms and applying this substitution, and both expressions will be identical.

E.g. Let's say there are two different expressions, P(x, y) and P(a, f(z)).

In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.

P(x, y) ... (I)

P(a, f(z)) ... (II)

Substitute x with a, and y with f(z) in the

first expression, and it will be represented as a/x and f(z)/y.

→ With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: [a/x, f(z)/y].

Conditions for Unification:

→ Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.

→ Number of arguments in both expressions must be identical.

→ Unification will fail if there are two similar variable present in the same expression.

For each pair of the following atomic sentences find the most general unifier (If exist).

1. Find the MGU of {p(f(a), g(Y)) and p(X, X)}

Soln:

$S_0$ => Here $\psi_1 = p(f(a), g(Y))$ and $\psi_2 = p(X, X)$

SUBST $\theta = \{f(a)/X\}$

$S_1$ => $\psi_1 = p(f(a), g(Y))$ and $\psi_2 = p(f(a), f(a))$

SUBST $\theta = \{f(a)/g(Y)\}$, unification failed.

Unification is not possible for these expressions.

2. Find the MGU of {p(b, x, f(g(z))) and ~~p(z, f(y), f~~ p(z, f(y), f(y))}

Soln:

Here, $\psi_1 = p(b, x, f(g(z)))$, and $\psi_2 = p(z, f(y), f(y))$

$S_0 \Rightarrow \{p(b, x, f(g(z))); p(z, f(y), f(y))\}$

SUBST $\theta = \{b/z\}$

$S_1 \Rightarrow \{p(b, x, f(g(b))); p(b, f(y), f(y))\}$

SUBST $\theta = \{f(y)/x\}$

~~$S_2 \Rightarrow \{p(b, f(y), f(g(b)); p(b, f(b)$~~

$S_2 \Rightarrow \{p(b, f(y), f(g(b))); p(b, f(g(b)), f(g(b))\}$

Unified successfully.

And Unifier = {b/z, f(y)/x, g(b)/y}.

3. Find the MGU of {p(x, x), and p(z, f(z))}

Here,

$\psi_1 = \{p(x, x)$, and $\psi_2 = p(z, f(z))$

$S_0 \Rightarrow \{p(x, x), p(z, f(z))\}$

SUBST $\theta = \{x/z\}$

$S_1 \Rightarrow \{p(z, z), p(z, f(z))\}$ Unification failed.

Hence, unification is not possible for these expressions.

4. Find the MGU of UNIFY(prime(11), prime(y))

Here, $\psi_1 = \{$prime(11), and $\psi_2 =$ prime(y)$\}$

$S_0 \Rightarrow \{$prime(11), prime(y)$\}$

SUBST $\theta = \{11/y\}$

$S_1 = \{$prime(11), prime(11)$\}$ Successfully unified.

Unifier : {11/y}

# Resolution in FOPL

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e. proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the Conjunctive normal form or clausal form.

Clause → Disjunction of literals (an atomic sentence) is called a clause. It is also known as a unit clause.

Conjunctive Normal form → A sentence represented as a conjunction of clauses is said to be Conjunctive normal form or CNF.

In FOPL, the process to ~~convert~~ apply the resolution method is as follows:

→ Convert the given axiom into CNF i.e. a conjunction of clauses. Each clause should be disjunction of literals.
→ Apply negation on the goal given.
→ Use literals which are required and Prove it

→ Unlike propositional logic, FOPL literals are complementary if one unifies with the negation of other literal.

Example of FOPL resolution:

Consider the following knowledge base:
1. Gita likes all kinds of food.
2. Mango and Chapati are food.
3. Gita eats almond and is still alive.
4. Anything eaten by anyone and is still alive is food.

Goal : Gita likes almond.
Solution: Convert the given sentences into FOPL as:

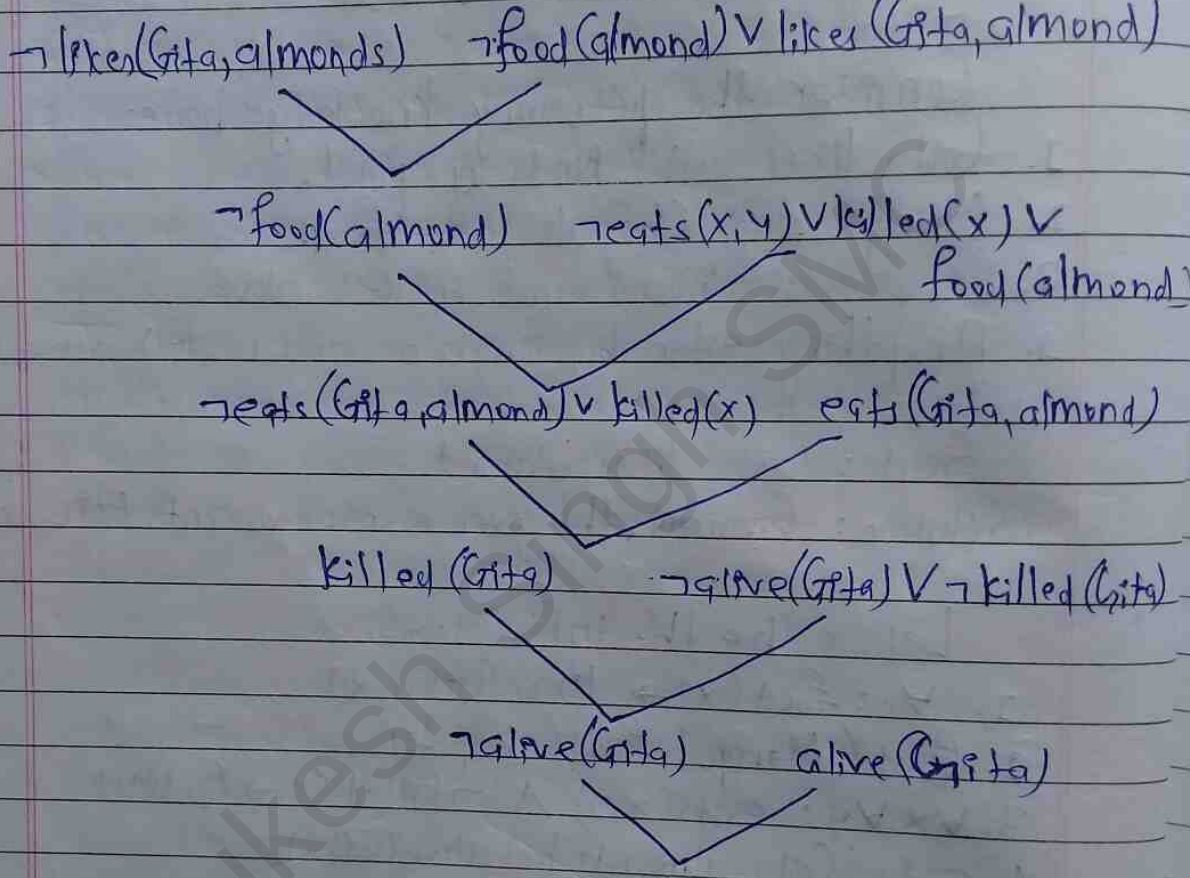
Let, x be the light sleeper.
1. $\forall x : food(x) \rightarrow likes(Gita, x)$
2. food(Mango), food(chapati)
3. $\forall x \forall y : eats(x,y) \wedge \neg killed(x) \rightarrow food(y)$
4. eats(Gita, almonds) ∧ alive(Gita)
5. $\forall x : \neg killed(x) \rightarrow alive(x)$
6. $\forall x : alive(x) \rightarrow killed(x)$

Goal : likes(Gita, almond)
Negated goal : ¬ likes(Gita, almond)

Now, rewrite in CNF form:
1. $\neg food(x) \vee likes(Gita, x)$
2. food(Mango), food(chapati)
3. $\neg eats(x,y) \vee killed(x) \vee food(y)$
4. eats(Gita, almonds), alive(Gita)
5. $killed(x) \vee alive(x)$

6. ¬alive(x) V ¬killed(x)

Finally construct the resolution graph:




Hence, we have achieved the given goal with the help of proof by Contradiction.
Thus it is proved that Gita likes almond.

Page No.
Date

# Forward Chaining and Backward Chaining in FOPL:

## A. Forward Chaining in FOPL:

Forward chaining in predicate Logic (FOPL) is different from forward chaining chaining in propositional logic. In FOPL, forward chaining is efficently implemented on first-order clauses. First-order clauses are the disjunction of literals of which exactly one is positive.

For example: $Crow(x) \wedge Thurst(x) \rightarrow Water(x)$.
$Crow(x)$
$Thurst(x)$.

Therefore, a definite clause is either atomic or an implication whose predecessor is a conjunction of positive literals. As a result, it results in a single positive literal.

Let's see an example of forward Chaining in FOPL:
Consider the below axioms:
1. Gita loves all types of clothes.
2. Suits are clothes.
3. Jackets are clothes.
4. Anything any wear and isn't bad is clothes.
5. Sita wears skirt and is good.
6. Renu wears anything Sita wears.

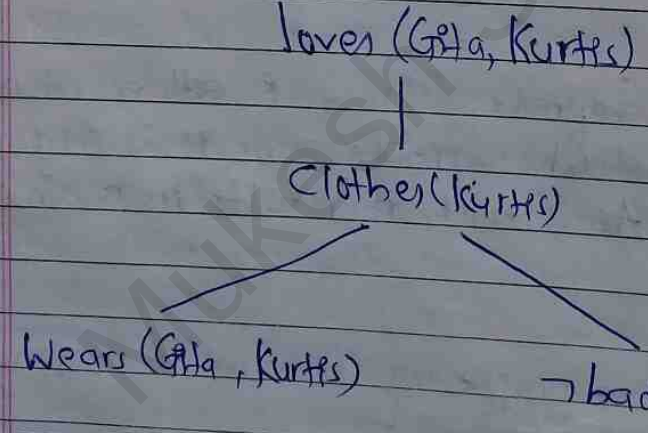
Apply forward Chaining and prove that Gita loves kurtis.

Solution: Convert the given axioms into FOPL as:

1. $\forall x$: Clothes(x) → loves(Gita, x).
2. Suits(x) → Clothes(x).
3. Jackets(x) → clothes(x)
4. Wears(x, y) ∧ ¬bad(y) → Clothes(x)
5. wears(Sita, skirt) ∧ → good(Sita)
6. Wears(Sita, x) → Wears(Renu, x)

To prove: Gita loves kurtis.
FOPL: loves(Gita, kurtis)

On applying forward chaining in the below graph:




Thus, it is proved that Gita loves kurtis.

Date

## Backward Chaining in FOPL:

In FOPL, backward chaining works from the backward direction of the goal, apply the rules on the known facts which could support the proof. Backward chaining is a type of AND/OR search because we can prove the goal by applying any rule in the knowledge base. A backward chaining algorithm is used to process the backward chaining.

### Example

Consider the following axioms:

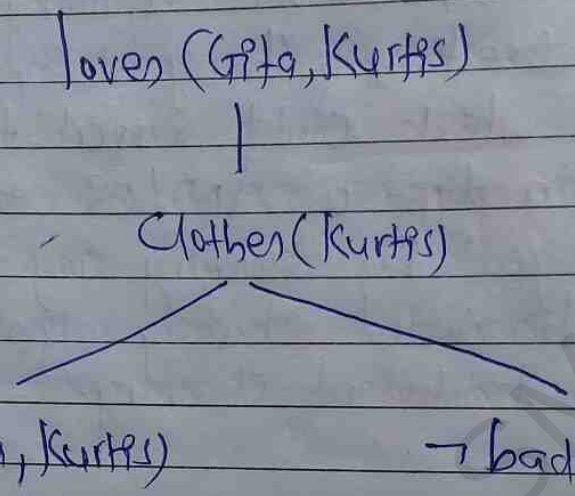
1. Gita loves all type of clothes.
2. Suits are clothes.
3. Jackets are clothes.
4. Anything any wear and isn't bad is clothes.
5. Sita wears skirt and is good.
6. Renu wear anything Sita wears.

Apply backward chaining and prove that Gita loves Kurtis.

1. $x$: clothes$(x) \rightarrow$ loves$(Gita, x)$.
2. Suits$(x) \rightarrow$ Clothes$(x)$
3. Jackets$(x) \rightarrow$ Clothes$(x)$.
4. wears$(x, y) \rightarrow \wedge \neg$bad$(y) \rightarrow$ clothes$(x)$
5. wears$(Sita, skirt) \wedge \neg$ good$(Sita)$
6. wears$(Sita, x) \rightarrow$ wears$(Renu, x)$.

To prove: Gita loves Kurtis.
FOPL: loves$(Gita, Kurtis)$.

Apply backward chaining in the below graph:




It is clear from the above graph, Gita wears Kurtis and does not look bad. Hence, Gita loves Kurtis.

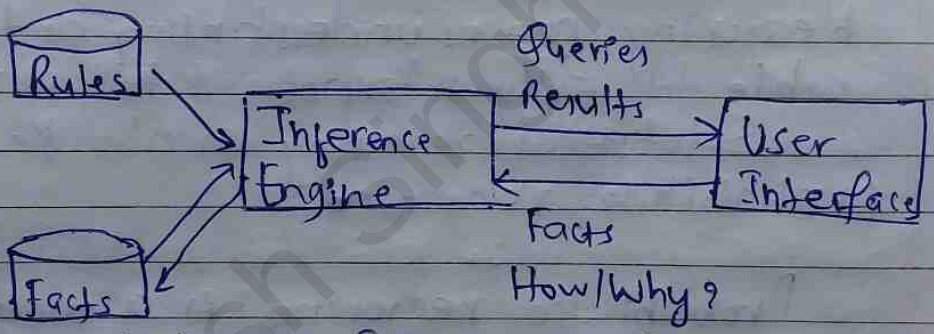
Note: We have seen that the graph of forward and backward chaining is same. It means that forward chaining follows the bottom-up approach and backward chaining follows the top-down approach.

## 3.4 Rule based deduction system

Rule based deduction system is a logical program that uses pre-defined rules to make deductions and choices to perform automated actions.

Rule based systems (RBS) provides automatic problem solving tools for capturing the human expertise and decision making.

Experts typically express most of their problem solving techniques in terms of antecedent-consequent rules.




Forward Chaining & backward chaining are the methods of Rule based system.

Properties of Rule based System:

→ They incorporate practical human knowledge in if-then rules.

→ They can solve a wide range of potentially complex problems by selecting relevant rules and then combining the results.

→ Determine the best sequence of rules to examine.

## 3.5 Statistical Reasoning - Probability and Bayes' theorem and causal networks, reasoning in belief network.

Statistical reasoning is the way people reason with statistical ideas and make sense of statistical information. Statistical reasoning may involve connecting one concept to another (e.g. center and spread) or may combine ideas about data and chance.

Reasoning means understanding and being able to fully interpret statistical results.

Statistical reasoning is also known as probabilistic reasoning.

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behaviour of someone for some situations," "A match between two teams or two players"

There are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

## Need of probabilistic reasoning in AI:

→ When there are unpredictable outcomes.
→ When specifications or possibilities of predicates becomes to large to handle.
→ When an unknown error occurs during an experiment.

In probabilistic reasoning there are two ways to solve problems with uncertain knowledge.
- Bayes's rule
- Bayesian statistics

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, we should understand some common terms:

## # Probability

Probability can be defined as a chance that an uncertain event will occur. It is the numerical method of the likelihood that an event will occur. The value of probability always remains 0 and 1 that represent ideal uncertainities.

$0 \leq P(A) \leq 1$, where $P(A)$ is probability of an event A.
$P(A) = 0$, indicates total uncertainity in an event A.
$P(A) = 1$, indicates total certainity in an event A.

We can find the probability of an uncertain event by using the below formula.

Probability of Occurence = $\frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$

- $P(\neg A)$ = probability of not happening event.
- $P(\neg A) + P(A) = 1$

Event → Each possible outcomes of a variable is called an event.

Sample Space → The collection of all possible events is called sample space.

Random Variables :→ Random variables are used to represent the events and objects in the real world.

Prior probability → The prior probability of an event is probability computed before observing new information.

Posterior Probability → The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional Probability
It is a probability of occuring an event when another event has already happened.

Let is suppose, we want to calculate the event A when event B has already occured, "the probability of A under the condition of B", it can be written as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Where, $P(A \cap B)$ = Joint probability of A and B.
$P(B)$ = Marginal probability of B.

If the probability of A is given and we have to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Example.
In a class, there are 70% of the students who like English and 40% of the students who like English and Mathematics, and then what is the percent of ~~Subjects~~ students those who like English also like Mathematics?

Solution:
Let, A is an event that a student likes Mathematics.
B is an event that a student likes English

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

# Bayes theorem

→ Bayes theorem is also known as Bayes rule, Bayes law or Bayesian reasoning, which determines the probability of an event with uncertain knowledge.

→ In probability, it relates the conditional probability and marginal probabilities of two random events.

→ Bayes theorem was named after the British mathematician, Thomas Bayes. The Bayesian inference is an application of Bayes theorem which is fundamental of Bayesian statistics.

→ It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

→ Bayes theorem allows the updating the probability prediction of an event by observing the new information of the real world.

Example: If cancer corresponds to one's age then by using Bayes theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes theorem can be derived using product rule and conditional probability of event A with known event B.

As from product rule, we can write,

$P(A \wedge B) = P(A|B)\,P(B)$ or - ①

Similarly, the probability of event B with known event A:

$$P(A \wedge B) = P(B|A)\,P(A) \quad \text{— (II)}$$

Posterior prob. (prob. of B when A is true)

Equating right hand side of both equations, we get

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \quad \text{— (I)}$$

Likelihood prob. evidence

Marginal probability (prob. of evidence)

Prior probability (probability of hypothesis)

→ The above equation (I) is called as Bayes rule or Bayes theorem. This equation is basic of most modern AI systems for probabilistic inference.

→ It shows the simple relationship between joint and conditional probabilities.

→ Here, $P(A|B)$ is known as posterior, which we need to calculate and it will be read as probability of hypothesis A where we have occured an evidence B.

→ $P(B|A)$ is called likelihood, in which we consider the hypothesis is true, then we calculate the probability of evidence.

→ $P(A)$ is called prior probability, probability of hypothesis before considering the evidence.

→ $P(B)$ is called marginal probability, pure probability of an evidence.

→ In the equation (I), in general, we can write $P(B) = P(A) * P(B|A)$, hence the Bayes rule can be written as:

$$P(A_i|B) = \frac{P(A_i) * P(B|A_i)}{\sum_{i=1}^{k} P(A_i) * P(B|A_i)}$$

Where $A_1, A_2, A_3 \ldots A_n$ is a set of mutually exclusive and exhaustive events.

## Applying Bayers rule:

Bayers rule/theorem allows us to compute the single term of P(A|B) and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayers rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})\, P(\text{cause})}{P(\text{effect})}$$

Example 1.

Q.1) What is the probability that a patient has diseases meningitis with a stiff neck?

Soln:

Given data.

A doctor is ~~always~~ aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

→ The known probability that a patient has meningitis disease is 1/30,000.

→ The known probability that a patient has a stiff neck is 2%.

Let A be the proposition that patient has stiff neck and B be the proposition that the patient has meningitis, so we can calculate the following as:

$P(A|B) = 0.8$

$P(B) = 1/30000$

$P(A) = 0.02$

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)} = \frac{0.8 \times \frac{1}{30000}}{0.02} = 0.0013333$$

Hence, we can assume that 1 patient out of 750 patients has meningitis disease with a stiff neck.

Q.2) From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King/Face), which means the drawn face card is a king card.

Solution:

$$P(King|Face) = \frac{P(Face|King) * P(King)}{P(Face)} \quad \text{— (1)}$$

P(King): Probability that a card is king = 4/52 = 1/13

P(Face): Probability that a card is face card = 3/13

P(Face/King): probability of a face card when we assume it is a king = 1

Putting all values in equation (1), we will get:

$$P(King|Face) = \frac{1 \times \frac{1}{13}}{\frac{3}{13}} = 1/3$$, it is a probability that a face card is a king card.

Application of Bayes theorem in AI:

→ It is used to calculate the next step of the robot when the already executed step is given.

→ Bayes theorem is helpful in weather forecasting.

→ It can solve the Monty Hall problem.

# Belief Network

"A Bayesian Belief Network is a probabilistic graphical model which represents a set of variable and their conditional dependencies using a directed acyclic graph." (DAG)

→ It is also called a Baye's network, belief network, decision network or Bayesian model.

→ Bayesian belief networks are probabilistic, because these networks are built from a probability distribution, and also we probability theory for prediction and anomaly detection.

→ It can be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction and decision making under uncertainty.

→ It is built from probability distribution.

→ It consists of two parts:

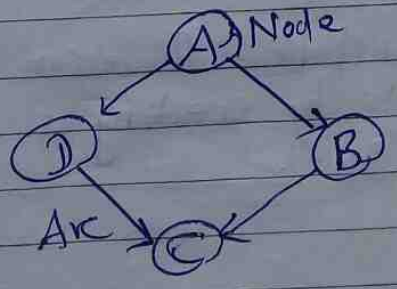
① Directed Acyclic Graph (DAG)

② Table of Conditional Probabilities (TCP)

A Bayesian belief network graph is made up of:

① Nodes → Corresponds to a random variable.

② Arcs (directed links) → represents casual relation or Cond. prob. among random variables



→ Each node corresponds to the random variables and variable can be continuous or discrete.

→ Arc or directed arrows represents the causal relationship or conditional probabilities between random variables.

- In the above diagram, A, B, C and D are random variables represented by the nodes of the network graph.
- If we are considering node B, which is connected with the node A by ~~the~~ a directed arrow, the node A is called the parent of Node B.
- Node C is independent of node A.

→ The Bayesian network has mainly two components:

① Casual Component
② Actual numbers

→ Each node in the Bayesian network has conditional probability distribution $P(X_i | Parent(x_i))$, which determines the ~~effect of that~~ the parent on that node.

→ Bayesian network is based on joint probability distribution and conditional probability.

Let us understand the Bayesian network through an example by creating a directed acyclic graph (DAG).

Page No.
Date

## Problem / Example

1) Calculate the probability that the alarm has sounded, but there is neither a burglary, nor an earthquake occured and David and Sophia both called the Harry.

### Solution:

→ The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.

→ The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.

→ The conditional distributions for each node are given as conditional probabilities table or CPT.

→ Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.

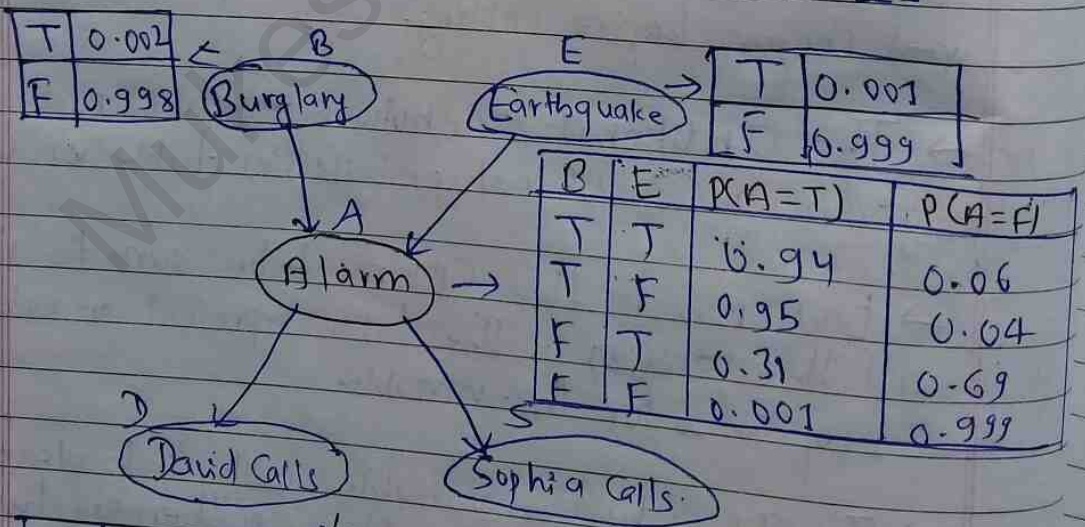
→ In CPT, a boolean variable with k boolean parents contains $2^k$ probabilities. Hence, if there are two parents, then CPT will contain 4 probability values.

List of all events occuring in this network:

- Burglary (B)
- Earthquake (E)
- Alarm (A)
- David Calls (D)
- Sophia Calls (S)

We can write the events of problem statement in the form of probability: $P[D, S, A, B, E]$, can rewrite the above probability statement using joint probability distribution:

$$P[D,S,A,B,E] = P[D \mid S,A,B,E] \cdot P[S,A,B,E]$$
$$= P[D \mid S,A,B,E] \cdot P[S \mid A,B,E] \cdot P[A,B,E]$$
$$= P[D \mid A] \cdot P[S \mid A,B,E] \cdot P[A,B,E]$$
$$= P[D \mid A] \cdot P[S \mid A] \cdot P[A \mid B,E] \cdot P[B,E]$$
$$= P[D \mid A] \cdot P[S \mid A] \cdot P[A \mid B,E] \cdot P[B \mid E] \cdot P[E]$$




| T | 0.002 |
|---|---|
| F | 0.998 |

| T | 0.001 |
|---|---|
| F | 0.999 |

| B | E | P(A=T) | P(A=F) |
|---|---|---|---|
| T | T | 0.94 | 0.06 |
| T | F | 0.95 | 0.04 |
| F | T | 0.31 | 0.69 |
| F | F | 0.001 | 0.999 |

| A | P(D=T) | P(D=F) |
|---|---|---|
| T | 0.91 | 0.09 |
| F | 0.05 | 0.95 |

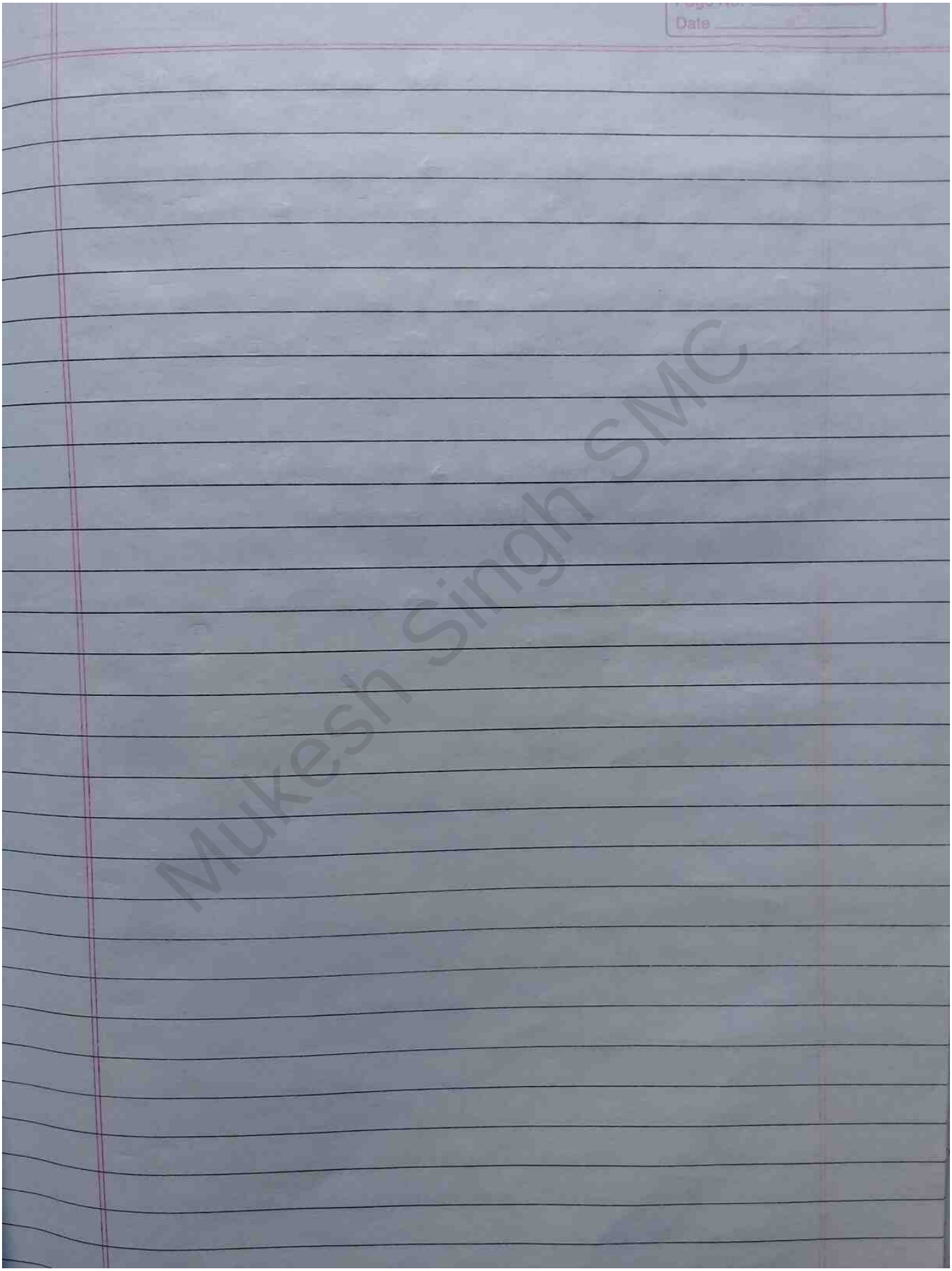
| A | P(S=T) | P(S=F) |
|---|---|---|
| T | 0.75 | 0.25 |
| F | 0.02 | 0.98 |

Lets take the observed probability for the Burglary and earthquake component:

P(B = True) = 0.002, which is the probability of burglary.
P(B = False) = 0.998, which is the probability of the burglary.
P(E = True) = 0.001, which is the prob. of minor earthquake.
P(E = false) = 0.999, which is the prob. that earthquake not occured.

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

P(S, D, A, ¬B, ¬E) = P(S|A) * P(D|A) * P(A|¬B ∧ ¬E) * * P(¬B) * P(¬E)
= 0.75 * 0.91 * 0.001 * 0.998 * 0.999
= 0.0006845.


Date
Mukesh Singh SMC

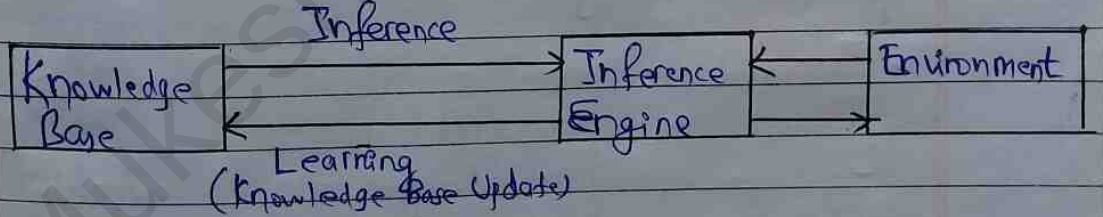
# Unit-4

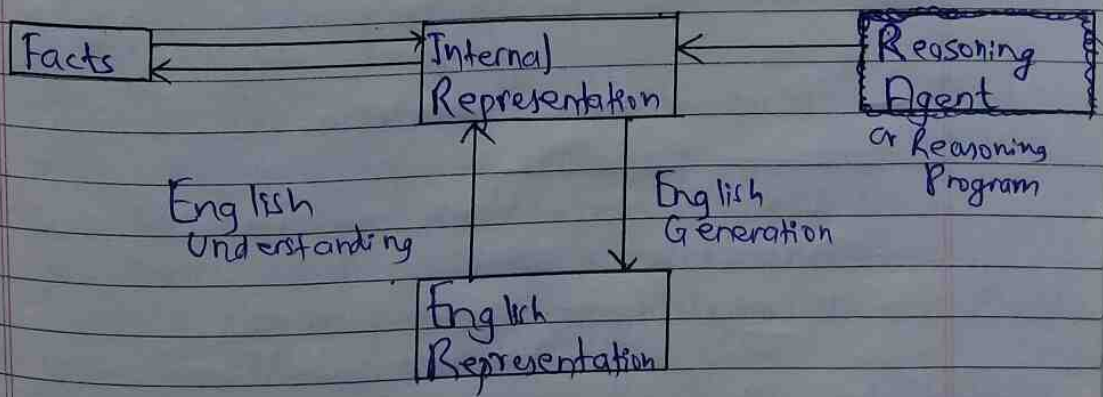
## Structured Knowledge Representation

Structured knowledge is the most basic knowledge and applied in problem solving. It tries to find out a relationship between concepts and objects.

### 4.1 Representations and Mappings

→ Mapping is the process that maps facts to representation and vice versa.
→ Forward mapping : facts to representation.
→ Backward mapping : representation to facts.



Page No.
Date

# Properties of Good Knowledge Representation:

i) Represential Adequacy → Ability to represent all kinds of knowledge needed in the domain.

ii) Inferential Adequacy → Ability to manipulate knowledge to derive new structures infered from old.

iii) Inferential Efficiency → Ability to perform inference in the most efficient directions.

iv) Acquisitional Efficiency → Ability to acquire new information easily.

## 4.2 Approaches to Knowledge Representation; Issues in Knowledge Representation

### Approaches to Knowledge Representation:

There are mainly four approaches to knowledge representation, which are given below:

1. Simple relational knowledge

It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.

This approach is famous in database systems where the relationship between different entities is represented.

Example

| Name | Weight | Age |
|---|---|---|
| Mukesh | 60 | 25 |
| Raj | 65 | 24 |
| Manjay | 48 | 26 |

2. Inheritable Knowledge

In this approach, all data must be stored into a hierarchy of classes.

All classes should be arranged in a generalized form or a hierarchical manner.

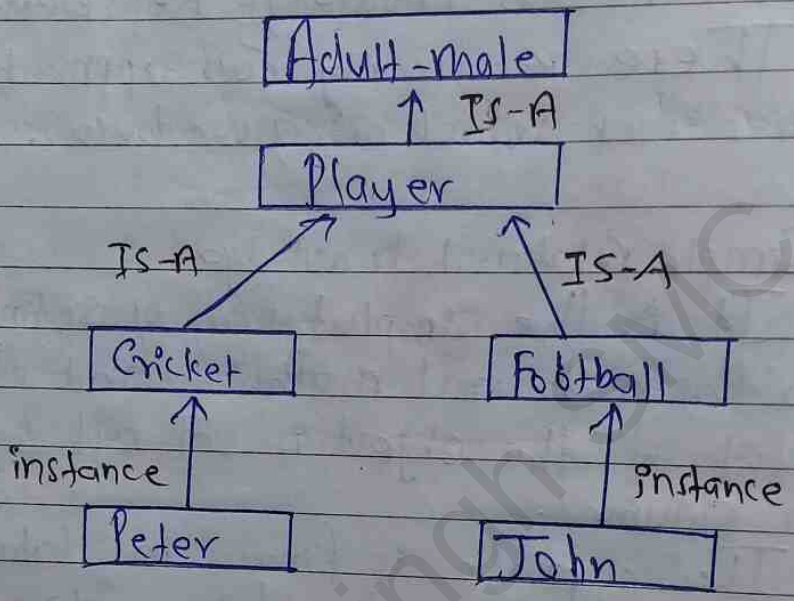
We apply inheritance property, elements inherit values from other members of a class.

Objects and values are represented in Boxed nodes.

We use arrows which point from objects to their

values.

Example




3. Inferential Knowledge

→ Inferential knowledge approach represents knowledge in the form of formal logics.

→ This approach can be used to derive more facts.

→ It guarantees correctness.

~~Example~~ Example

Let is suppose there are two statements:

a. Marcus is a man.

b. All men are mortal.

Then it can be represented as:

man (Marcus)

$\forall x = man(x) \dashrightarrow mortal(x)$s

## 4. Procedural Knowledge

→ Procedural knowledge aproach we small programs and codes which describes how to do specific things, and how to proceed.
→ In this approach, one important rule is used which is If-Then rule.
→ In this knowledge, we can use various coding languages such as LISP language and Prolog language.
→ We can easily represent heuristic or domain-specific knowledge using this approach.
→ But it is not necessary we can represent all cases in this approach.

## # Issues in knowledge representation

The fundamental goal of knowledge Representation is to facilitate inferencing (conclusion) from knowledge.
The issues that arise ~~during~~ while using KR techniques are many. Some of them are explained below:

(i) Important attributes
Are any attributes of objects so basic that they have been occured in almost every problem domain?

(ii) Relationship among attributes
Are there any important relationships that exist among attributes of objects?

iii) Choosing Granularity
At what level should knowledge be represented?

iv) Set of Objects
How should sets of objects be represented?

v) Finding right structure
Given a large amount of knowledge stored, how can relevant parts be accessed when they are needed?

4.3 Semantic nets, frames

Semantic nets (or Semantic network)

Semantic network is an alternative to ~~known~~ predicate logic as a form of knowledge representation.

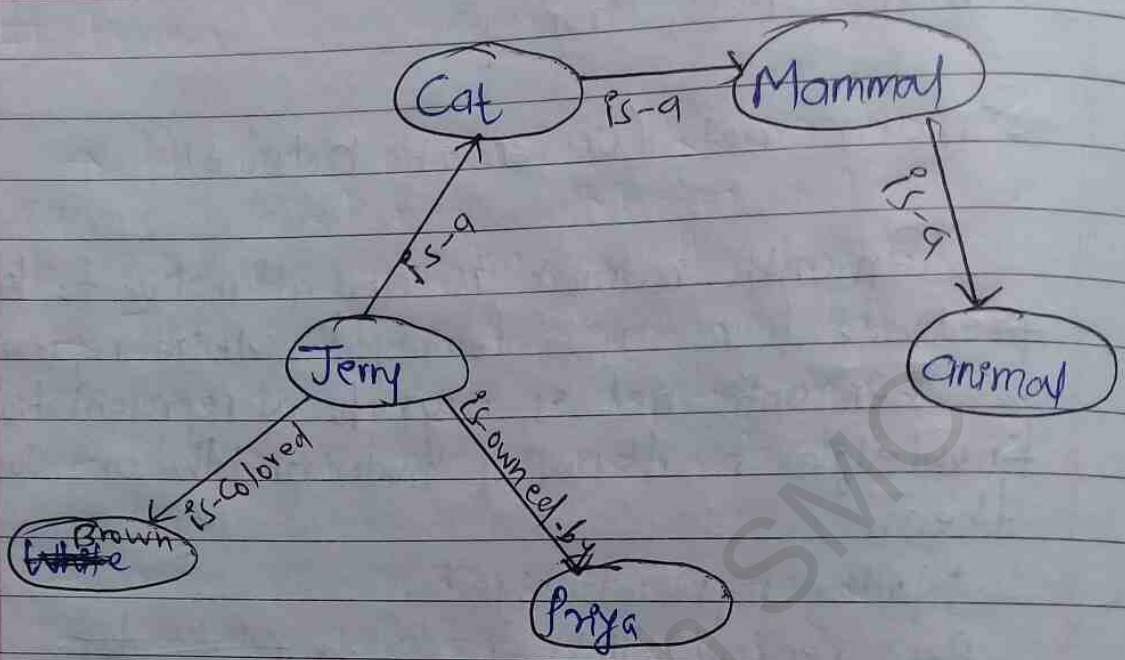
Semantic net is a graphical representation of knowledge in terms of nodes and the arcs connecting them.

Nodes represent objects.
Arcs represent links or edges. ~~or relationships.~~
The links are used to express relationships.

This representation consists of mainly two types of relations:
a. IS-A relation (inheritance)
b. Kind-of-relation (relates one class to another).

Example

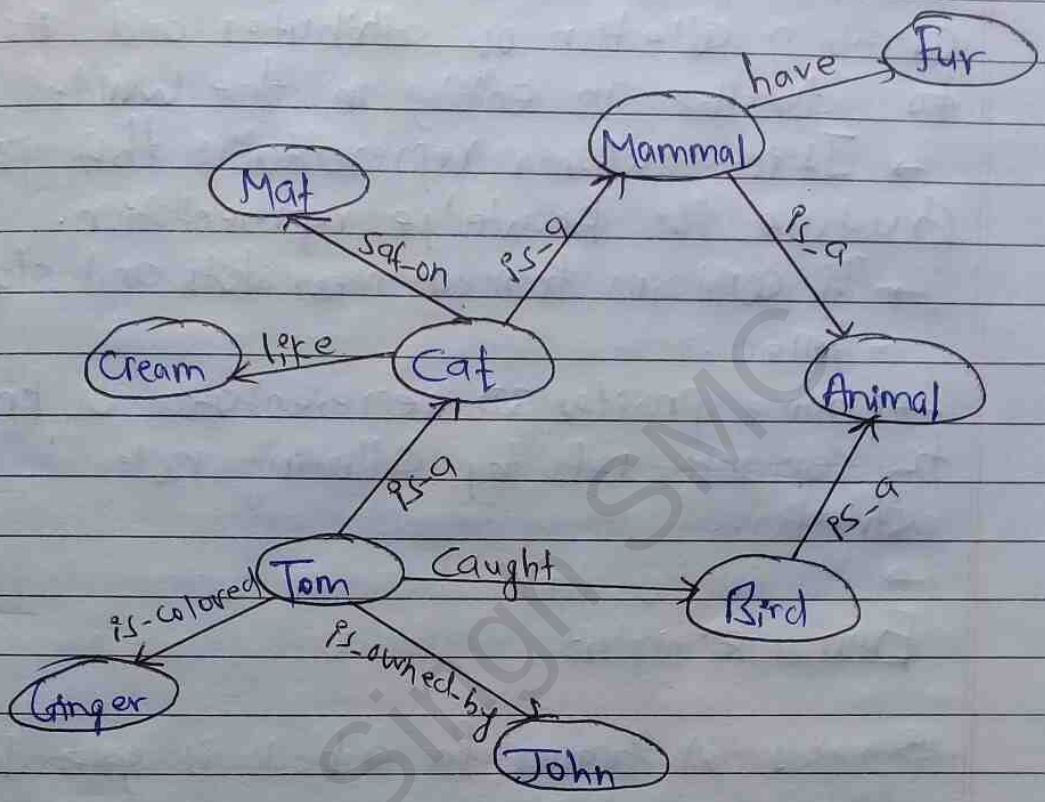
Represent the following data in Semantic Nets:
a. i) Jerry is a cat.
ii) Jerry is a mammal
iii) Jerry is owned by Riya.
iv) Jerry is brown coloured.
v) All mammals are animals.




## Example 2

i) Tom is a cat.
ii) Tom caught a bird.
iii) Tom is owned by John.
iv) Tom is ginger in colour.
v) Cats like cream.
vi) The cat sat on the mat.
vii) A cat is a mammal.
viii) A bird is an animal.
ix) All mammals are animals.
x) Mammals have fur.

It is argued that this form of representation is closer to the way human's structure knowledge.




Advantages of Semantic Nets:

~~# of Frames~~ → Natural and modular.

→ Efficient, simple and understandable.

→ Translatable to PROLOG without ~~efficiency~~ difficulty.

→ Meaning is simple to generate.

Disadvantages of Semantic Nets:

→ No standard definition.

→ Not intelligent.

→ Depends on the designer.

→ Creates confusion on same network created by multiple people.

# Frames

→ A frame is a record like structure which consists of collection of attributes and its values to describe an entity in the world.
→ It is a schema representation that provides structure for knowledge representation.
→ It contains frame name, slots and slot fillers.
→ Frame provides 3D representation of knowledge to semantic nets by allowing nodes to have structure.

Frame examples:

Example 1: A frame for a book is given below:

| Slots | Fillers |
|---|---|
| Title | Artificial Intelligence |
| Genre | Computer Science |
| Author | Peter Norving |
| Edition | Third Edition |
| Year | 1996 |
| Pages | 1152 |

Example 2: Lets suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So, following is the frame representation for this:

| Slots | Fillers |
|---|---|
| Name | Peter |
| Profession | Doctor |
| Age | 25 |
| Marital status | Single |
| Weight | 78 |

Advantages of frame representation:
→ It makes the programming easier by grouping the related data.
→ It is flexible and used by many applications in AI.
→ It is very easy to add slots for new attributes.
→ It is easy to understand and visualize.

Disadvantages of frame representation:
→ Inference mechanism is not easily processed.
→ It has a much generalized approach.

## 4.4 Conceptual dependencies and scripts

### Conceptual dependency (CD):

→ CD theory was developed by Schank in 1973 to 1975 to represent the meaning of Natural Language (NL) sentences.
- It helps in drawing inferences.
- It is independent of the language.

→ CD represents the knowledge acquired from natural language input.
→ If two or more sentences are identical in meaning, there should be only one representation.

→ Sentences are represented as a series of diagrams depicting actions.
→ Agents and objects are represented.

Primitive Acts of CD theory:

- ATRANS → Transfer of an Abstract relationship. (e.g. give)
- PTRANS → Transfer of the physical location of an object. (e.g. go)
- ~~PROPEL~~ →
- MTRANS → Transfer of mental information. (e.g. tell)
- MBUILD → Construct new information from old. (e.g. decide)
- PROPEL → Application of physical force to an object.

(e.g. push).

- MOVE → Movement of a body part by its owner. (e.g. punch, kick).
- GRASP → Grasping of an object by an action (e.g. throw)
- INGEST → Ingesting of an object by an animal. (e.g. eat)
- EXPEL → Expulsion of something from the body of an animal (e.g. cry)
- SPEAK → Producing of sounds. (e.g. say)
- ATTEND → Focusing of a sense organ towards a stimulus. (e.g. listen).

~~There~~

# Conceptual Category

There are four conceptual categories:

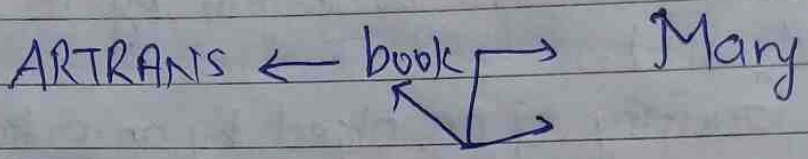
i) ACT → Actions (one of the CD primitives)
ii) PP → Objects (picture producers)
iii) AA → Modifiers of actions (action aiders)
iv) PA → ~~Modifications of~~ Modifiers of PP's (picture aiders)

Example of CD:

Sentences:

1. Mary take a book from John.
2. John gave Mary a book.

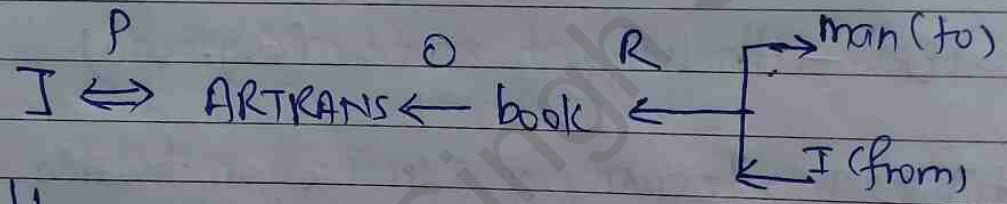
Conceptual dependency representation:




Example 2

I gave a book to the man.

CD representation is as follows:




Here,

→ Arrow indicates the 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.

# Scripts

→ Script is a structure which is used to represent the knowledge.

→ It is a structured representation describing a sequence of events in a particular context.

→ Scripts are frame like structures used to represent commonly occuring experiences such as going to movies, shopping in market, eating in resturant, banking, etc.

→ A script consists of slots and information (knowledge) contained in it.

Various components of scripts are:

- Script Name → Title
- Track → Special situation, specific variation
- Roles → people involve in the event described in script.
- Entry condition → required pre situation to execute the script.
- Props → Non live objects involve in the Script.
- Scenes → The actual sequence of events that occur.
- Result → Condition that will be true after execution of script.

Example of Script:

Script Name : Shopping Script
Track : Super Market
Roles : Shopper, daily attendant, check out clerk, cashier.
Entry Condition : Shopper needs groceries, food market open
Props : Shopping cart, Display aisles, market items, money.

Scene 1: Enter Market
Shopper PTRANS shopper into Market.
Shopper PTRANS shopping cart to Shopper.

Scene 2: Shop for item
Shopper MOVE shopper through aisles.
Shopper ATTEND eyes to display items.
Shopper PTRANS item to shopping cart.

Scene 3: Check Out
Shopper MOVE Shopper to checkout stand
Shopper ATTEND eyes to charges.
Shopper ATRANS money to cashier.
Sacker ATRANS bag to shopper.

Scene 4: Exit Market
Shopper PTRANS shopper to exit market.

Results: Shopper has less money.
Shopper has grocery items
Market has less grocery items
Market has more money

Example 2:
Script for going to the bank to withdraw money.

Script: Withdraw Money
Track: Bank
Roles: P= Customer, E= Employee, C= Cashier
Entry Condition: P has no or less money,

The bank is open,
Props : Money, Counter, form, Token.

Scene 1 : Entering
P PTRANS P into the bank
P ATTEND eyes to E.
P MOVE P to E.

Scene 2 : Filling form
P MTRANS signal to E.
E ATRANS form to P
P PROPEL form for writing
P ATRANS form to P
E ATRANS form to P

Scene 3 : Withdrawing Money
P ATTEND eyes to Counter
P PTRANS P to queue at the counter
P PTRANS token to C
C ATRANS money to P.

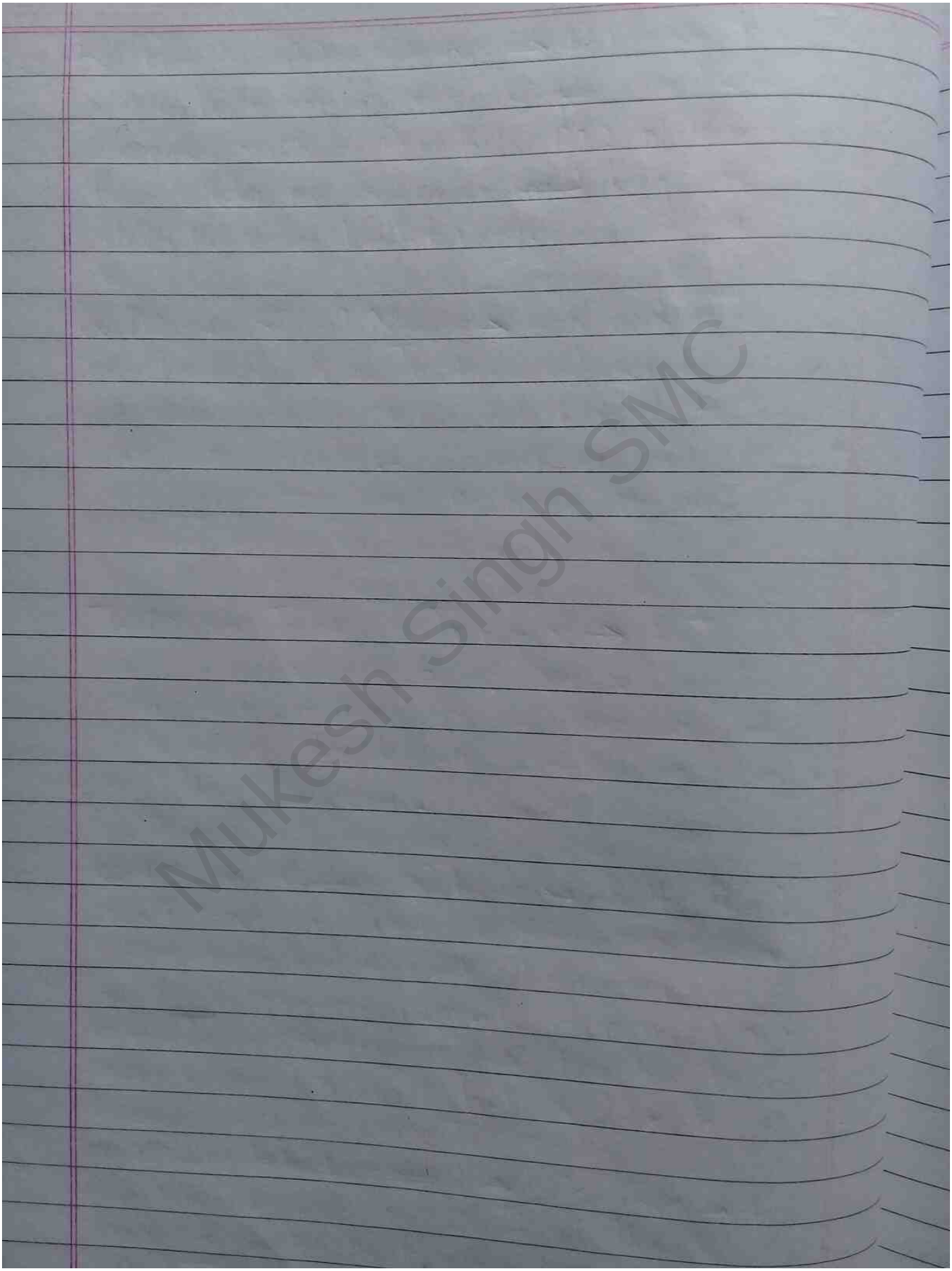
Scene 4 : Exiting the bank
P PTRANS P out of bank

Results : P has more money

Date

Advantages of Scripts:
→ Ability to predict events.
→ A single coherent interpretation may be build up from a collection of observations.

Disadvantages of Scripts:
→ Less general than frames.
→ May not be suitable to represent all kinds of knowledge.



# Unit-5

## Application of AI System in Education

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Health Care, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

1. AI in Astronomy → AI can be very useful to solve complex universe problems.

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

3. AI in Gaming → The AI machines can play strategic games like chess, where the machines needs to think of a large number of possible places.

4. AI in Finance → The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security → AI can be used to make your data more ~~same~~ and secure. Some examples such as AEG bot, AI2 platform, are used to determine ~~Sytware~~ ~~and~~ bug and cyber attacks in a better way.

6. AI in Social Media → Social media ~~sites~~ 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.

7. AI in Travel & Transport → AI is capable of doing various travel related works such as from making travel arrangement to ~~Suggesting~~ the hotels, flights, and best routes to the customers.

8. AI in ~~Autom~~ Robotics → With the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed. Humanoid robots named as Erica and Sophia has been developed which can talk and behave like humans.

9. AI in ~~Entertainment~~ → Education → AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.

AI in future can be a work as a personal

Page No.
Date

Virtual tutor for students, which will be accessible easily at any time and any place.

## 5.1 Expert Systems (Architecture, Expert System development process)

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. The system helps in decision making for complex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

The performance of ES is based on the expert's knowledge stored in its knowledge base (KB). The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

# Architecture / Block diagram of an Expert System:

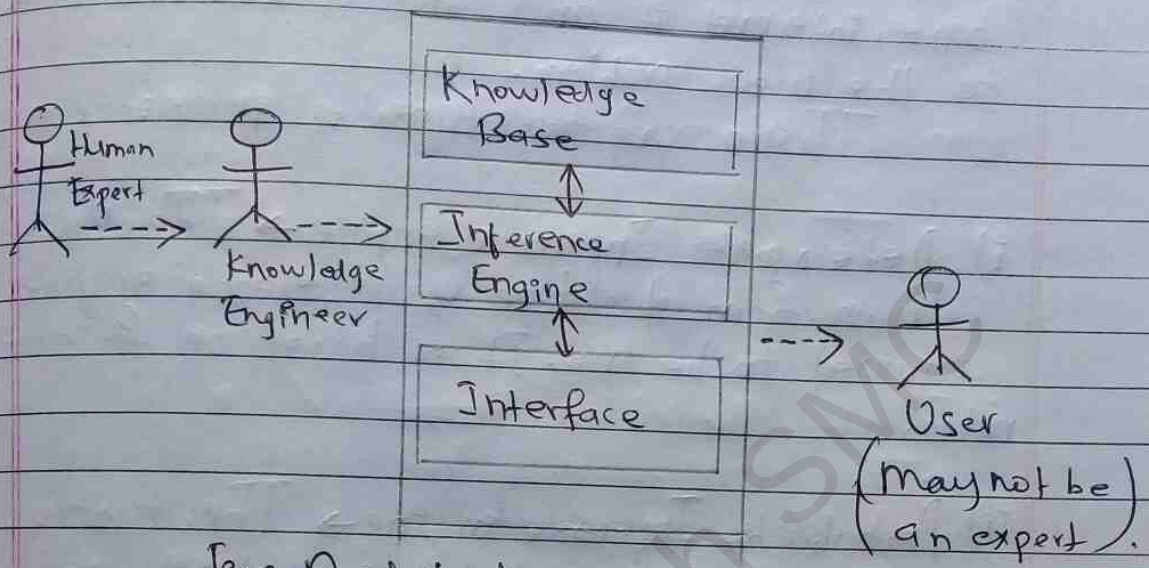



Fig: Architecture of an Expert System.

## 3. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an ~~expert~~ input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

## 2. Inference Engine (Rule engine)

The inference engine is known as the brain of the 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.

The function of the inference engine is to fetch the relevant knowledge from the knowledge base, interpret it and find a solution relevant to the users problem.

There are two type of inference engines:

i) Deterministic Inference engine → The conclusion drawn from this type of inference engine are assumed to be true. It is based on facts and rules.

ii) Probabilistic Inference engine → This type of inference engine contains uncertainity in conclusions and based on the probability.

Inference engine uses the below modes to derive the solutions:

a) Forward Chaining → It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.

b) Backward Chaining → It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. Knowledge Base

The knowledgebase is a type of storage that stores knowledge from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the expert system.

It is similar to a database that contains information ~~about~~ and rules of a particular domain or subject.

One can view the knowledgebase as collection of objects and their attributes. Such as a lion is object and its attributes are: it is a mammal, it is not a domestic animal, etc.

Types of knowledge:

a) Factual knowledge → The knowledge which is based on facts and accepted by knowledge engineers come under factual knowledge.

b) Heuristic knowledge → This knowledge is based on practice, the ability to guess, evaluation, and experience.

Components of knowledge base:

i) Knowledge Representation → It is used to formalize the knowledge stored in the knowledge base using the if-else rules.

ii) Knowledge Acquisition → It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the

knowledge from various experts, and store that knowledge into knowledge base.

## Expert System Development Process:

The process of ES development is iterative. Steps in developing ES include:

i) Identify ~~the~~ Problem ~~Definition~~ Domain
→ The problem must be suitable for an expert system to solve it.
→ Find the experts in ~~the~~ task domain for the project.
→ Establish cost-effectiveness of the system.

ii) Design the system
→ Identify the ES technology.
→ Know and establish the degree of integration with the other systems and databases.
→ Realize how the concepts can represent the domain knowledge best.

iii) Develop the Prototype
→ ~~Identify the ES technology.~~
→ From knowledge base, the knowledge engineer works to:
- Acquire domain knowledge from the expert.
- Represent it in the form of If-then-else rules.

iv) Test and Refine the Prototype
→ The knowledge engineer uses sample case to test the prototype for any deficiencies in performance.
→ End users test the prototype of the ES.

v) Develop and Complete the ES
→ Test and ensure the interaction of the ES with all elements of its environment including end users, data-
→ base and other information systems.
→ Document the ES project well.
→ Train the user to use ES.

vi) Maintain the System
→ Keep the knowledge base up-to-date by regular review and update.
→ Cater for new interfaces with other information systems, as well as those systems ~~above~~ evolve.

OR

## The Process of Building an Expert System:

i) Determining the Characteristics of the problem.

ii) Knowledge engineer and domain expert work in Coherence to define the problem.

iii) The knowledge engineer translates the knowledge into a computer-understandable language. He designs an inference engine, a reasoning structure, which can use knowledge when needed.

iv) Knowledge expert also determines how to integrate the use of uncertain knowledge in the reasoning process and what type of explanation would be useful.

## 5.2 Application of Expert System in Education

Applications of Expert System:

1) Design Domain → Camera lens design, automobile design.

2) Medical Domain → Diagnosis systems to deduce cause of disease from observed data, conducting medical operations on humans.

3) Monitoring Systems → Comparing data continuously with observed system or with prescribed behaviour such as leakage monitoring in long petroleum pipeline.

4) Process Control Systems → Controlling a physical process based on monitoring.

5) Knowledge Domain → Finding out faults in vehicles, computers.

6) Finance/ Commerce → Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

# Applications of Expert System in Education

i) Recognizing student characteristics
ii) Student performance analysis
iii) Prediction of student performance
iv) Basic evaluation of student competence.
v) Evaluation of academic programs.
vi) Digital Library
vii) Quality improvement of learning
viii) Lesson Plan
ix) Efficiency of teaching and learning process
x) E-learning evaluation

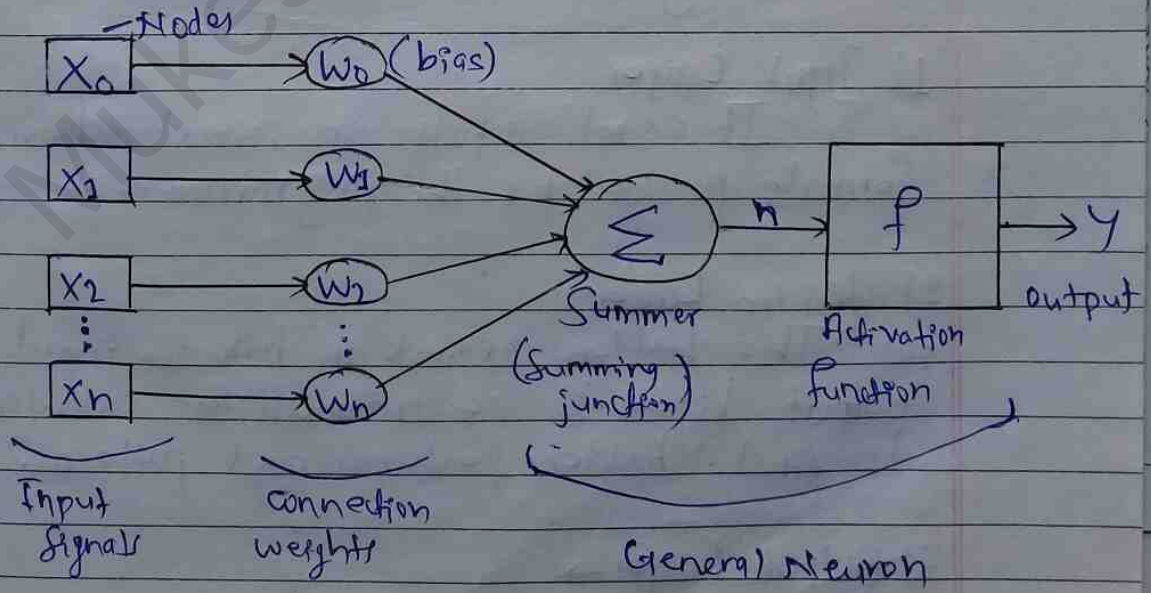
5.3 Neural Network (Mathematical Model, g of realizati-on, Network Structure)

An Artificial Neural Network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and process a information.

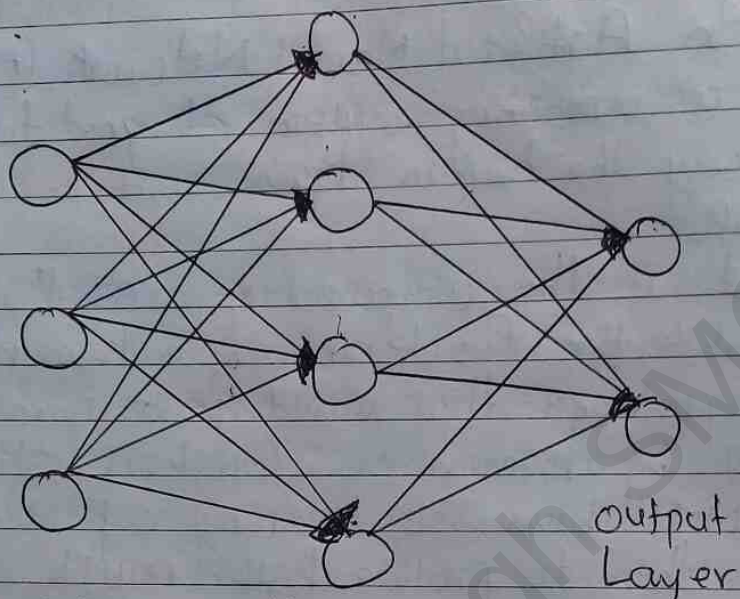
It is the component of AI that is meant to simulate the functioning of a human brain. It solves problems that would prove impossible or difficult by human or statistical standards.

ANNs have self learning capabilities that enable them to produce better results as more data becomes available.

Mathematical Model of Neural Network:



Basic Structure of ANN (neural network):




Artificial Neural Network primarily consists of three layers:

1) Input Layer.
It accepts inputs in several different formats provided by the programmer.

2) Hidden Layer
The hidden presents in between input and output layers. It performs all the calculations to find hidden features and patterns.

3) Output Layer→
The input goes through a series of transformations using the hidden layer, which finally results in

output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function (or summing function).

$$\sum_{i=1}^{n} W_i + x_i + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation function choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

## 5.4 Application of Neural Network in education

Overall applications of Neural Networks:

i) Aerospace → Autopilot aircrafts, aircraft fault detection.

ii) Automotive → Automobile ~~guiding~~ guidance systems.

iii) Military → Weapon orientation and steering, target tracking, facial recognition, signal / image identification.

iv) Electronics → Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.

v) Financial → Real estate appraisal, loan advisor, mortgage screening.

vi) Industrial → Manufacturing process control, product design and analysis, paper quality prediction, project planning and management.

vii) Medical → Cancer cell analysis, EEG and ECG analysis, prosthetic design.

viii) Speech → Speech recognition, speech classification, text to speech conversion.

ix) Telecommunication → Image and data

compression, automated information services, real-time spoken language translation.

x) Transportation → Truck Brake System diagnosis, vehicle scheduling, routing systems.

xi) Software → Pattern recognition in facial recognition, optical character recognition etc.

xii) Applications of Neural Network in education:

## 5.5 Natural Language Processing (Steps of NLP, parsing)

NLP stands for Natural Language Processing, which is a part of Computer Science, Human Language and Artificial Intelligence.

It is the technology that is used by machines to understand, analyze, manipulate, and interpret humans's languages.

It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction and topic segmentation.

### Components of NLP:

1. Natural Language Understanding (NLU)

NLU helps the machines to understand and analyze human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations and semantic rules.

NLU involves following tasks:

→ It is used to map the given input into useful representation.

→ It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

NLG acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning and Text Realization.

Steps of NLP:

There are following five phases of NLP:

1. Lexical Analysis ⇒

This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences and words.

2. Syntactic Analysis (Parsing)

Syntactic analysis is used to check grammar, word arrangements and shows the relationship among the words.

Example → Agra goes to the Poonam.

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases and sentences.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

Pragmatic is the fifth and the last phase of NLP. It helps you to discover the intended

effect by applying a set of rules that characterize cooperative dialogues.

For example: "Open the door" is interpreted as a request instead of an order.

## 5.6 Application of NLP in education

Overall applications of NLP:

i) Question Answering

Question answering focuses on building systems that automatically answer the question asked by humans in a natural language.

ii) Spam Detection

Spam detection is used to detect unwanted e-mails getting to user's inbox.

iii) Sentiment Analysis

Sentiment analysis is also known as opinion mining. It is used on the web to analyze the behaviour, attitude and emotional state of the sender.

iv) Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

Example: Google Translate

v) Spelling Correction

Microsoft Corporation provides word processor software like MS-Word, Powerpoint for the spelling correction.

vi) Speech Recognition

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

vii) Chatbot

It is used by many companies to provide the customers chat services.

viii) Information extraction

It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

ix) Natural Language Understanding (NLU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

Applications of NLP in education:

## 5.7 Basic Concept of Machine Learning and Versioning

Machine learning is a subset of AI which provides machines the ability to learn automatically and improve from experience without being explicitly programmed.

In the sense, it is the practice of getting machines to solve problems by gaining the ability to think.

The term "machine learning" was first introduced by Arthur Samuel in 1959. We can define it in a summarized way as:

"Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed."

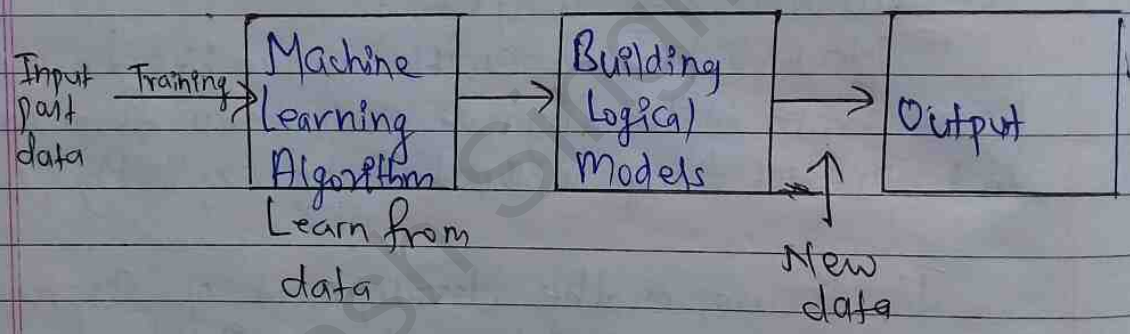
It focuses on the development of computer programs that can access ~~and~~ data and use it to learn for themselves.

The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop, conventional algorithms to perform the needed tasks.

## How does Machine learning work?

Machine learning System learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.




Suppose, we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm.

Classification of Machine Learning:

At broad level, machine learning can be classified into three types:

1. Supervised Learning

Supervised learning is a type of machine learning method in which we provide some labded data to the machine learning system in order to train it, and on that basis, it predicts the output.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on the supervision, and it is same as when a student learns things in the supervision of the teacher. The example of supervised learning is spam filtering.

Supervised learning can be grouped further into two categories of algorithms:

i) Classification

ii) Regression

2 Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labelled, classified or categorized, and the algorithm needs to act on that data without any supervision.

The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have predetermined result. The machine tries to find ~~the~~ useful insights from

the huge amount of data. It can be further classified into two categories of algorithms:
i) Clustering
ii) Association

3. Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance.

In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

# Machine Visioning

Machine vision is a technology that enables automatic inspection and analysis for applications including automatic inspections, process control, and robotic guidance by using image processing.

Machine vision is a technical capability that is integrated with existing technologies in new ways and applies it with the aim to solve real-world problems.

Page No.
Date

Machine vision is a systematic engineering discipline and can be considered distinct from computer vision which is a form of computer science and not done through a tangible piece of hardware such as a vision box or camera attached to a robot.

Machine vision is the body of the system and computer vision is the intelligence of the system, similar to how a computer is a frame for what goes inside such as the computer chips that power up the computer.

## 5.8 Application of Machine Learning in education

Overall applications of Machine Learning:

Below are some most trending real world applications of Machine Learning:

1. Image Recognition

It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is Automatic friend tagging suggestion. Whenever we upload a photo with our facebook friends, then we automatically get a tagging suggestion with name and the technology behind this is machine learning's face detection and recognition algorithm.

2. Speech Recognition

While using Google, we get an option of "Search by voice", it comes under speech recognition and it's a popular application of machine learning.

Speech recognition is the process of converting voice instructions into text, and it is also known as "Speech to text".

Examples: Google assistant, Siri, Cortana and Alexa.

3. Traffic Prediction:

- If we want to visit a new place, we take help of Google Maps, which shows the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving or heavily congested with the help of two ways:
→ Real time location of the vehicle from Google Map
→ Average time taken on past days at the same time.

4. Product Recommendations
Machine learning is widely used by various e-commerce and entertainment companies such as Amazon, Netflix, etc. for product recommendation to the user. Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.

5. Self-driving Cars
It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering
We always receive an important email in our inbox with important symbol and spam emails into our spam box, and the technology behind this is Machine Learning.

7. Medical Diagnosis
In medical science, Machine Learning is used for diseases diagnosis. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain. It helps in finding brain tumors and

other brain related diseases easily.

8. Automatic Language Translation

The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.