C.V RAMAN POLYTECHNIC
BHUBANESWAR

6th Semester

STUDY MATERIAL

Artificial Intelligence & ML

(AI&ML)

Prepared By:

Ipsita Ankita Hota

Artificial Intelligence

UNIT-1

1.1 Definition of AI & History of AI

Definition of AI:

Artificial Intelligence (AI) is the simulation of human intelligence in machines that are designed to think, learn, and solve problems. AI includes subfields such as machine learning, deep learning, natural language processing, and computer vision.

History of AI:

- **1950s:** Alan Turing proposed the Turing Test to determine if a machine can exhibit intelligent behavior.
- **1956:** John McCarthy coined the term "Artificial Intelligence" at the Dartmouth Conference.
- **1960s-70s:** Early AI research focused on rule-based systems and symbolic reasoning.
- **1980s:** Expert systems and machine learning techniques gained popularity.
- **1990s-2000s:** AI progressed with advancements in deep learning and natural language processing.
- **2010s-Present:** AI is widely used in real-world applications such as self-driving cars, healthcare, and robotics.1.2 Goals and Applications of AI

Goals of AI:

• Automation: Reduce human effort in repetitive tasks.
• Efficiency: Improve speed and accuracy of decision-making.
• Adaptability: Learn from data and improve over time.
• Problem-solving: Tackle complex problems in different domains.

Applications of AI:

Healthcare:

1. **Helping Doctors See Inside the Body Better:** AI is like a super helper for doctors when they look at pictures of the inside of a patient's body, like X-rays or MRIs. It uses smart algorithms to find things like problems, tumors, or broken bones very accurately.
2. **Developing Medications Quickly and Cost-Effectively:** AI acts like a super scientist in the lab. It uses certain algorithms to predict how different chemicals can fight diseases. This helps us make new medicines much quicker and at a low cost.

3. **Detecting Health Problems Early:** AI acts as a health detective. It looks at your health information to find out if you might get certain diseases in the future. When it sees a high risk, doctors can step in early to help you stay healthy. This is really important for conditions like diabetes and heart problems because catching them at this time means better treatment and less trouble for the patient.

Finance:

1. **Identifying and Prevention of Fraud:** AI keeps an eye on bank transactions all the time. They act like super detectives who can spot strange things happening with money, like someone using a credit card in a weird way. When they see something fishy, they raise the alarm and help the bank stop bad people from stealing money.
2. **Automated Trading:** AI helps a skilled trader who works automatically. It uses various algorithms to swiftly buy and sell stocks while analyzing all the market information.

Social Media:

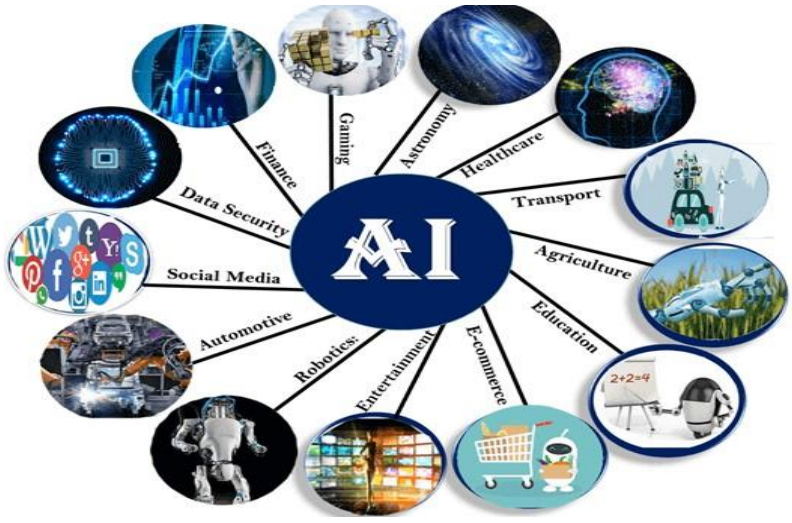
1. **Smart Suggestions:** AI helps as a guide on social media. It watches what you like and what you do, and then it suggests things you might enjoy, like posts, videos, or ads. It acts as someone who knows your tastes and shows you stuff you're really into, making your social media experience more enjoyable and personalized.
2. **Trend Analysis:** AI keeps track of all the chats and what's popular right now. This helps companies and regular folks understand what everyone's thinking and talking about. It acts as a social media news reporter that keeps customers in the loop about what's hot and what people are buzzing about.

Robotics:

1. **Self-Moving Robots:** AI makes robots really smart at moving around on their own. It's like giving them a built-in GPS and a clever brain.
2. **Object Recognition and Manipulation:** AI gives robots sharp eyes and clever hands. It helps them see objects clearly and then pick them up and move them just right.

Gaming:

1. **Smart Game Characters:** AI is like the brains behind game characters that players don't control.
2. **Making Games Look and Feel Real:** AI helps to make games look and act more like the real world. They create graphics that look just like the things we see, and they make how things move in games feel realistic, like in real life.



APPLICATION OF AI in Different Sector

1.3 Intelligent Agent

An intelligent agent is a system that perceives its environment through sensors and acts upon it using actuators to achieve specific goals. An agent can be:

Human-Agent: A human agent has eyes, ears, and other organs that work for sensors and the hand, legs, and vocal tract work for actuators.

Robotic Agent: A robotic agent can have cameras, an infrared range finder, NLP for sensors and various motors for actuators.

Software Agent: Software agents can have keystrokes, file contents as sensory input and act on those inputs, and display output on the screen.

An intelligent agent is an autonomous entity that acts upon an environment using sensors and actuators to achieve goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be rational.

Types of Intelligent Agents:

• Simple Reflex Agents: Respond directly to perceptions (e.g., thermostat).
• Model-Based Agents: Maintain an internal state to make better decisions.
• Goal-Based Agents: Act to achieve specific objectives.
• Utility-Based Agents: Optimize actions to maximize utility.
• Learning Agents: Improve performance based on past experiences.

1.4 Computer Vision

Computer Vision (CV) is a field of AI that enables machines to interpret and analyze visual data (images and videos). Computer Vision works by enabling computers to interpret and process visual data through a series of steps:

1. **Image Acquisition** – Capturing images or videos using cameras or sensors.
2. **Pre-processing** – Enhancing image quality (e.g., noise reduction, resizing).
3. **Feature Extraction** – Identifying key patterns like edges, textures, and shapes.
4. **Object Detection & Recognition** – Detecting objects and classifying them.
5. **Decision Making** – Using AI models to interpret the data and take action.

It uses techniques like **image processing, machine learning, and deep learning (CNNs, YOLO, Faster R-CNN)** to perform tasks like face recognition, medical imaging, and autonomous driving.

Applications of Computer Vision:

• Facial Recognition: Used in security systems and social media.
• Medical Imaging: Helps detect diseases like cancer in X-rays.
• Autonomous Vehicles: Detects pedestrians, traffic signals, and obstacles.
• Industrial Inspection: Quality control in manufacturing.
• Augmented Reality: Enhances real-world experiences in gaming and navigation.

1.5 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a branch of AI that enables machines to understand, interpret, and generate human language.

It works through the following key steps:

1. **Text Acquisition** – Collecting text data from sources like documents, social media, or speech-to-text conversion.
2. **Text Preprocessing** – Cleaning and preparing text by:
    - Tokenization (splitting text into words/sentences)
    - Stopword removal (removing common words like "the", "is")
    - Lemmatization & Stemming (reducing words to their root form)
3. **Feature Extraction** – Converting text into numerical representations using:
    - Bag of Words (BoW)
    - TF-IDF (Term Frequency-Inverse Document Frequency)
    - Word Embeddings (Word2Vec, GloVe, BERT)
4. **Understanding & Processing** – Applying Machine Learning (ML) or Deep Learning (DL) models to analyze text:
    - Sentiment Analysis (detecting emotions)
    - Named Entity Recognition (NER) (identifying names, locations)
    - Machine Translation (e.g., Google Translate)
    - Text Summarization & Chatbots
5. **Text Generation & Response** – Producing human-like text using models like GPT, Transformers, and LSTMs.

NLP combines **linguistics, statistics, and deep learning** to power applications like chatbots, search engines, voice assistants, and translation services.

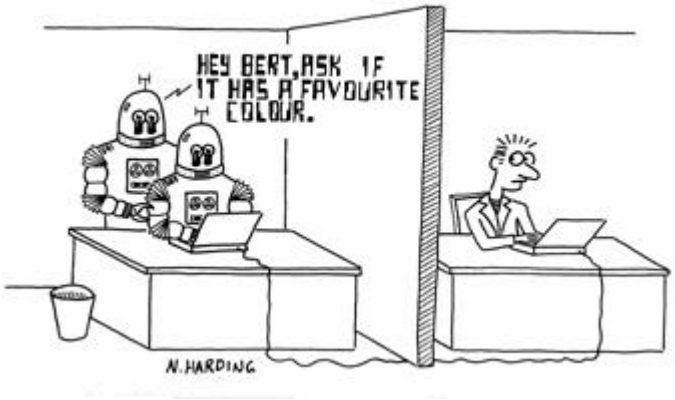
Applications of NLP:

• Chatbots & Virtual Assistants: Alexa, Siri, Google Assistant.
• Machine Translation: Google Translate, DeepL.
• Sentiment Analysis: Analyzing opinions on social media and reviews.
• Speech Recognition: Converting spoken language into text (e.g., voice commands).
• Text Summarization: Extracting key information from large documents.

1.6 Turing Test

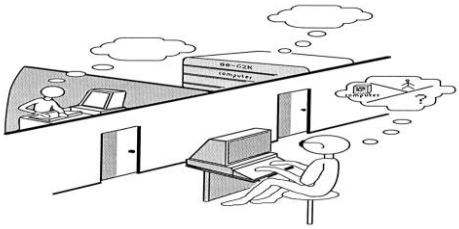
Turing always believes in Thinking Machine and he has proposed a research paper in 1950 having Title “Computing Machinery and Intelligence” (Turing, 1950), where he stated the famous Turing Test. In this Test he has proposed “The Imitation Game”, where three players are involved out of which one player is a man (called as A), another is a woman (called as B) and the third player is the Interrogator (called as C and could be of any sex). The interrogator(C) is confined in a room and connected through a communication link with (A) and (B). (Turing, 1950) The comic picture shown below elaborates the Imitation Game of Alan Turing. Now if interrogator asking any queries to either

A and B, then answer comes from the other side which cannot be recognized by interrogator that whether the answer coming from A or from B. In this situation if we replace either A or B with a computer, then this test will be known as The Turing Test.



This Comic picture is borrowed from https://plus.maths.org/content/os/issue5/turing/index

Here the interrogator asking queries to the other parties and if
interrogator fails to identify the actual party who is answering the queries then the Turing Test is successful.



1.7 Problem-solving in games

Problem-solving games are activities that require players to use critical thinking skills to solve puzzles. Example activities include escape rooms, Sudoku, and murder mysteries. The purpose of these exercises is to sharpen reasoning and decision-making skills in group settings and to do team building with employees. Problem-solving in games refers to the process of overcoming challenges, making strategic decisions, and optimizing actions to achieve a goal. This applies to both **game design (AI, mechanics)** and **gameplay (player strategies, puzzles, missions).**

Unit-2

2.1 Search, Search Space, and Search Tree

Search: The process of finding a path or solution from a starting point to a goal state within a given problem space. It is widely used in AI for tasks like pathfinding, game playing, and decision-making.

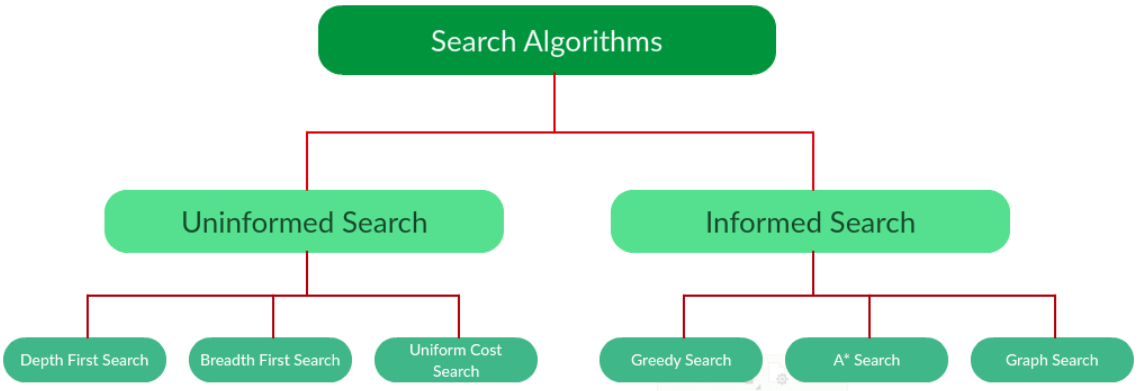
Search Space: The search space consists of all possible states or configurations that the algorithm can explore to find a solution. It defines the range of possibilities that need to be evaluated.

Search Tree: A search tree is a representation of the search process where:

- Nodes represent different states in the search space.
- Edges represent actions that transition one state to another.
- The root node is the starting state, and goal nodes represent solutions.

Example: In a maze, each position is a node, and moving up, down, left, or right forms the edges of the search tree.

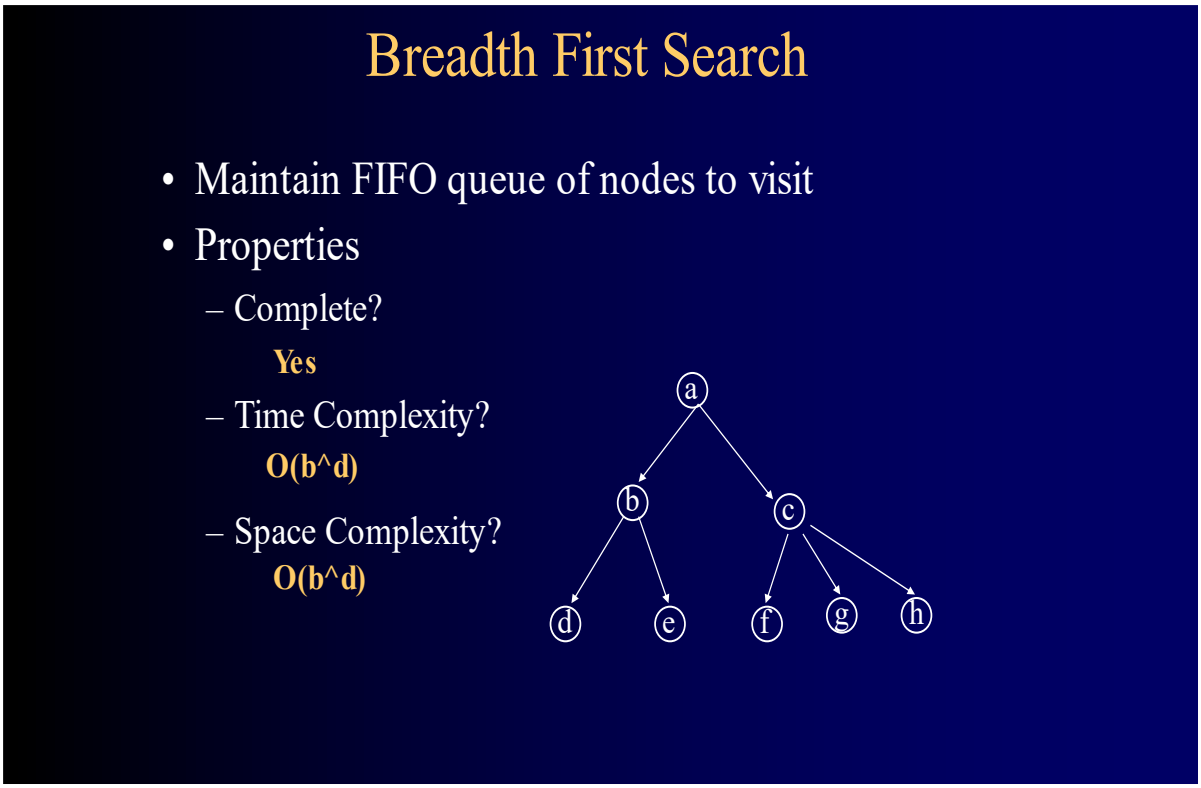
2.2 Categories and Types of Search



Search algorithms are categorized into two main types:

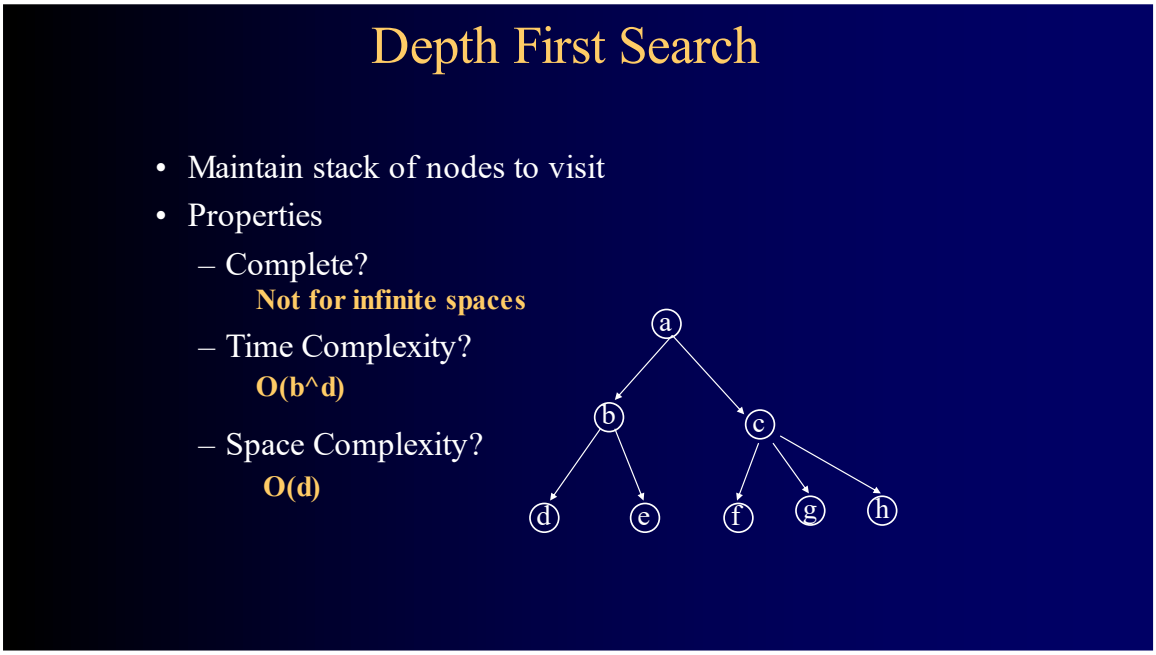
1. Uninformed (Blind) Search: These algorithms explore the search space without prior knowledge about the goal. They rely only on the problem definition and systematically explore possibilities.

- Breadth-First Search (BFS): Explores all possible states at the current depth before moving deeper. Guarantees finding the shortest path.



OUTPUT: abcdefgh

- **Depth-First Search (DFS):** Explores as deep as possible in one branch before backtracking. Efficient in memory but may not always find the shortest solution.



OUTPUT: abdecfgh

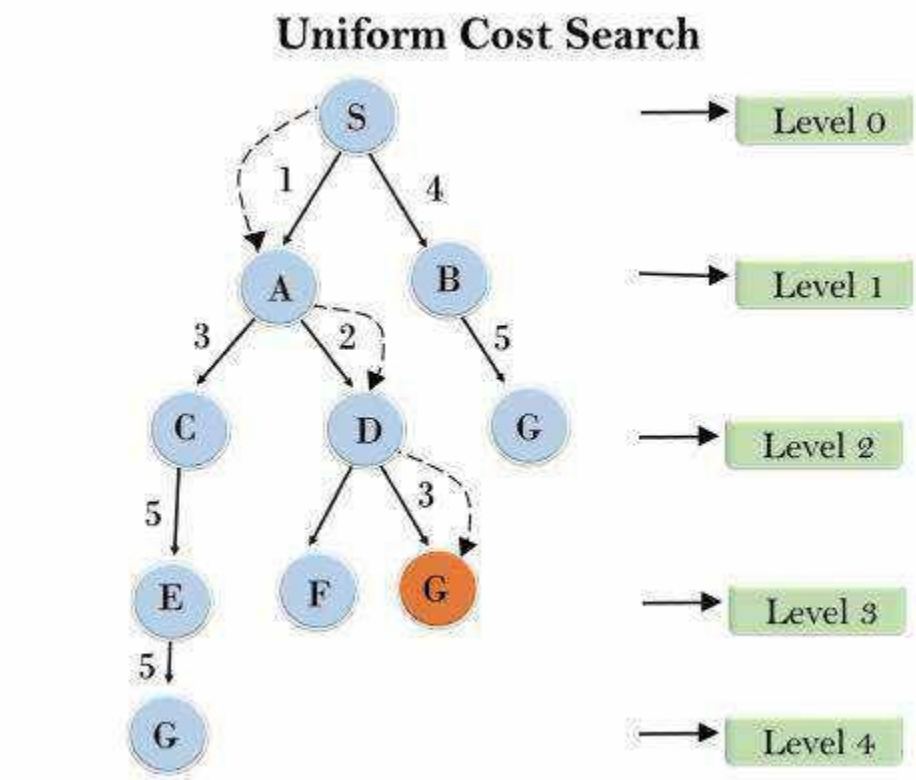
Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Advantages: o Uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages: o It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

Example:



Completeness:

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

Time Complexity:

Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

Hence, the worst-case time complexity of Uniform-cost search is $\mathbf{O(b^{1 + \lfloor C^*/\varepsilon \rfloor})}$/.

Space Complexity:

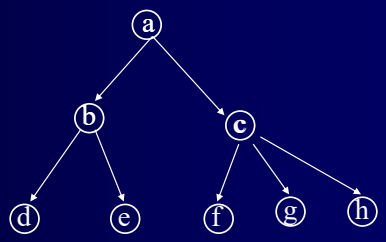
The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is $\mathbf{O(b_{1 + \lfloor C^*/\varepsilon \rfloor})}$.

Optimal:

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Properties
  - Complete?
    Yes
  - Time Complexity?
    O(b^d)
  - Space Complexity?
    O(d)



OUTPUT: abcabdecfgh

2. Informed (Heuristic) Search: These algorithms use heuristics (rules of thumb) to guide the search towards the goal more efficiently.

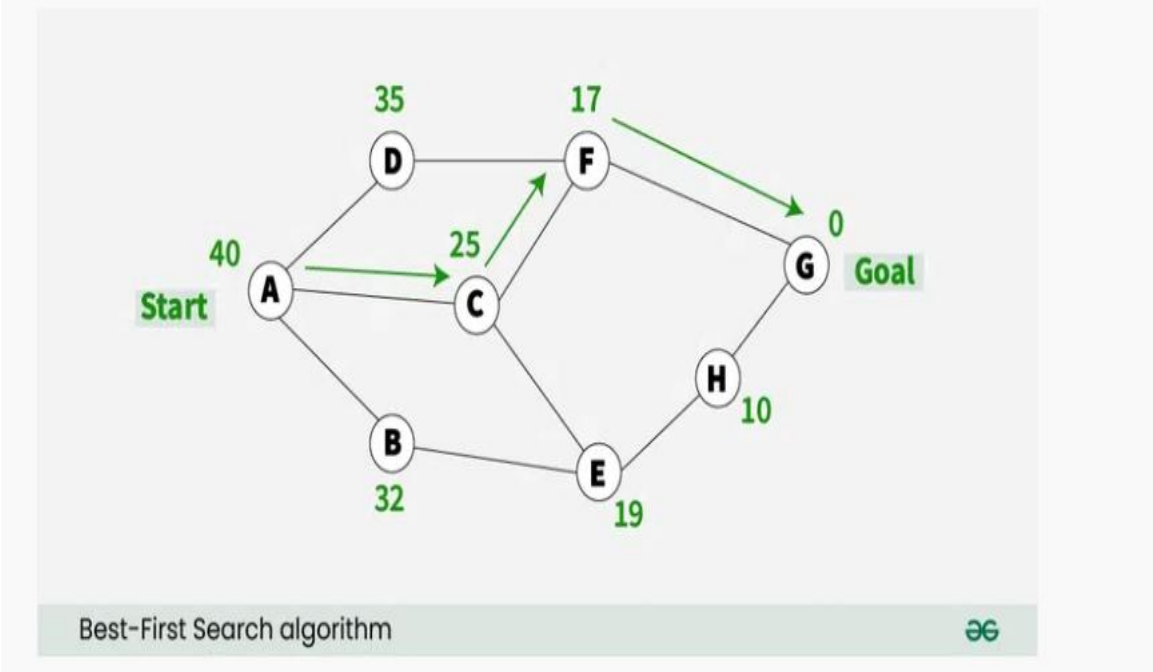
- Greedy Best-First Search:

Expands the node that appears closest to the goal based on a heuristic function.

- Greedy Best-First Search works by evaluating the cost of each possible path and then expanding the path with the lowest cost. This process is repeated until the goal is reached.
- The algorithm uses a heuristic function to determine which path is the most promising.
- The heuristic function takes into account the cost of the current path and the estimated cost of the remaining paths.
- If the cost of the current path is lower than the estimated cost of the remaining paths, then the current path is chosen. This process is repeated until the goal is reached.
-

An example of the best-first search algorithm is below graph, suppose we have to find the path from A to G

The values in red color represent the heuristic value of reaching the goal node G from current node



1) We are starting from A , so from A there are direct path to node B(with heuristics value of 32) , from A to C (with heuristics value of 25) and from A to D(with heuristics value of 35) .
2) So as per best first search algorithm choose the path with lowest heuristics value , currently C has lowest value among above node . So we will go from A to C.

3) Now from C we have direct paths as C to F(with heuristics value of 17) and C to E(with heuristics value of 19) , so we will go from C to F.

4) Now from F we have direct path to go to the goal node G (with heuristics value of 0) , so we will go from F to G.

5) So now the goal node G has been reached and the path we will follow is ***A->C->F->G*** *.*

Advantages of Greedy Best-First Search:

- **Simple and Easy to Implement:** Greedy Best-First Search is a relatively straightforward algorithm, making it easy to implement.
- **Fast and Efficient:** Greedy Best-First Search is a very fast algorithm, making it ideal for applications where speed is essential.
- **Low Memory Requirements:** Greedy Best-First Search requires only a small amount of memory, making it suitable for applications with limited memory.
- **Flexible:** Greedy Best-First Search can be adapted to different types of problems and can be easily extended to more complex problems.
- **Efficiency:** If the heuristic function used in Greedy Best-First Search is good to estimate, how close a node is to the solution, this algorithm can be a very efficient and find a solution quickly, even in large search spaces.
-

Disadvantages of Greedy Best-First Search:

- **Inaccurate Results:** Greedy Best-First Search is not always guaranteed to find the optimal solution, as it is only concerned with finding the most promising path.
- **Local Optima:** Greedy Best-First Search can get stuck in local optima, meaning that the path chosen may not be the best possible path.
- **Heuristic Function:** Greedy Best-First Search requires a heuristic function in order to work, which adds complexity to the algorithm**.**
- **Lack of Completeness:** Greedy Best-First Search is not a complete algorithm, meaning it may not always find a solution if one is exists. This can happen if the algorithm gets stuck in a cycle or if the search space is a too much complex.
-

A* Algorithm: Uses both the path cost and heuristic value to find the optimal solution efficiently.

- ***A* (A Star) Algorithm***

- The A* algorithm was proposed by Hart, Nilsson and Raphael.

- This algorithm is the combined form of *Branch and Bound*, *Dijkstra's algorithm* and *Best First Search.*

- The Best First Search calculates cost in forward direction whereas B&B only calculates past cost.

- In other word, if ***f(n)=g(n) + h(n)*** where n is the current state and ***f(n)*** is total cost from *Start* to *Goal*.
- So, f(n) estimates the cost of the path from *Start* to *Goal* via the state ***n***. (i.e. Start n Goal)
- If the above ***evaluation function*** used with best first search, then it will be called as ***algorithm A***.
- Again if the heuristic value h(n) ≤ the cost of minimal path from n to Goal, Then the ***algorithm A*** will be called as ***"A* Algorithm".***
- Note that, all A* algorithms are admissible. *(i.e. admissibility of A* algorithm is a theorem)*

Algorithm

Procedure A*()

{

Open List := { Start }

Closed List := { }

f(start):= h(start)

Path(start):= Empty

While Open ≠ $Empty$

{

transfer the node n from Open to Closed list, where f(n) has minimum cost among all elements of Open list.

IF (n == Goal)

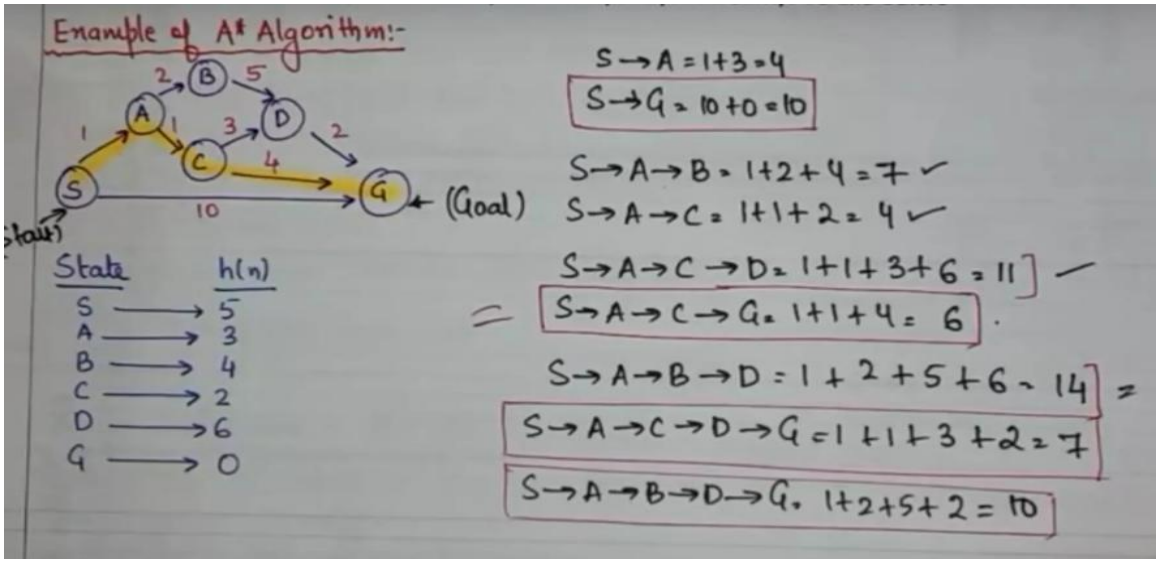
return Path.

Generate the child nodes of n.

For each child node(c) of n **do**

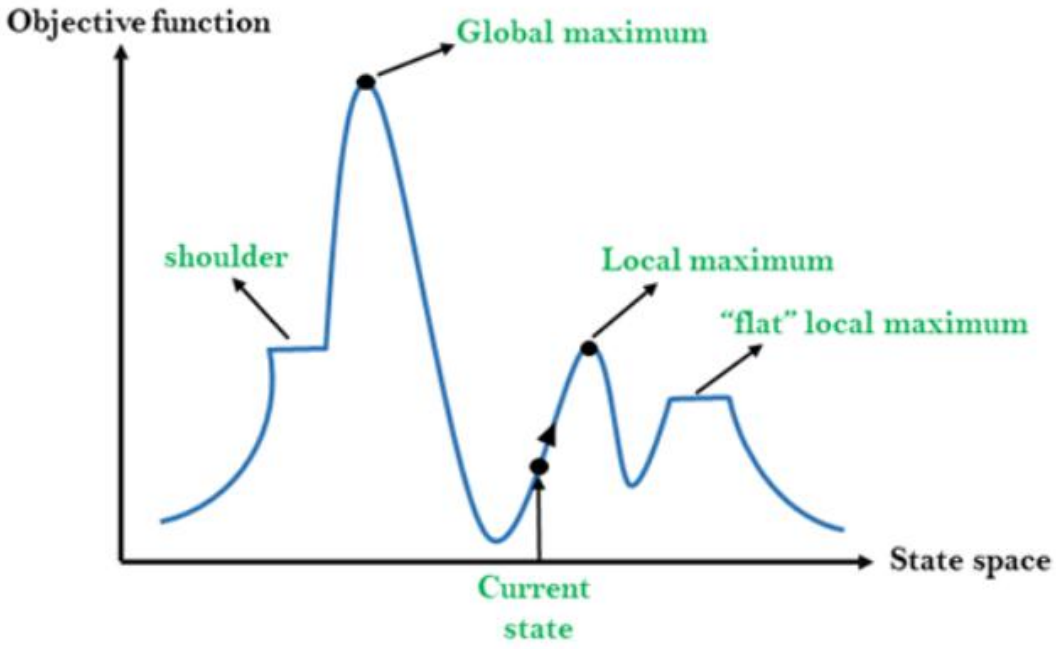
```
        IF ( c is not available in both Open and Closed List )
            add ( c )  to Open list and set Path(c) := Closed List.
            g( c )  :=  number of edge covered to reach c from start.
            f( c ) := g ( c ) + h( c )
        IF ( c is in Open List )
            IF ( the current g ( c ) > previous value of g ( c ))
                set Path(c) := Closed List.
                g ( c ) := previous value of g(c)
                f (c ) := g (c) + h(c)
        IF ( c is in Closed List )
            IF ( the current g (c ) > previous value of g (c))
                set Path(c) := Closed List.
                g ( c ) := previous value of g(c)
                f (c ) := g (c) + h(c)

END For
        }
    Return Failure;
    }
```



Hill Climbing:

Moves towards increasing values of a heuristic function, like climbing a hill to find the peak.



Different regions in the state space landscape:

Local Maximum: Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

Global Maximum: Global maximum is the best possible state of state space 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.

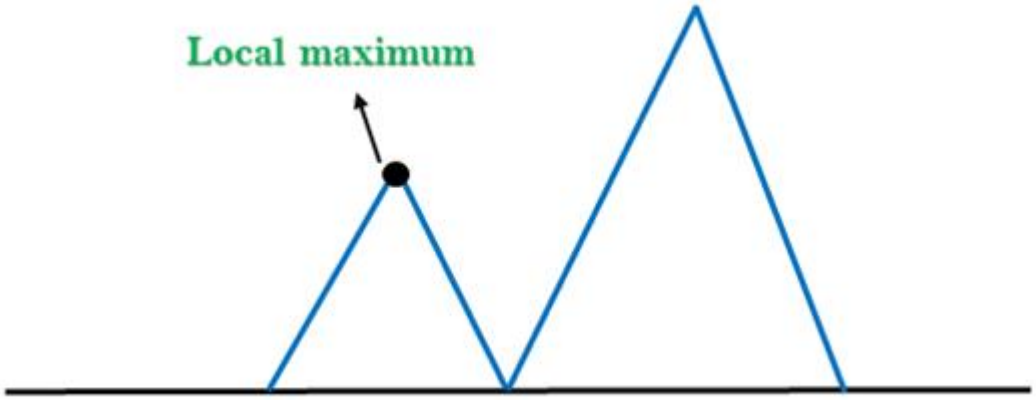
Flat local maximum: It is a flat space in the landscape where all the neighbor states of current states have the same value.

Shoulder: It is a plateau region which has an uphill edge.

Problems in Hill Climbing Algorithm:

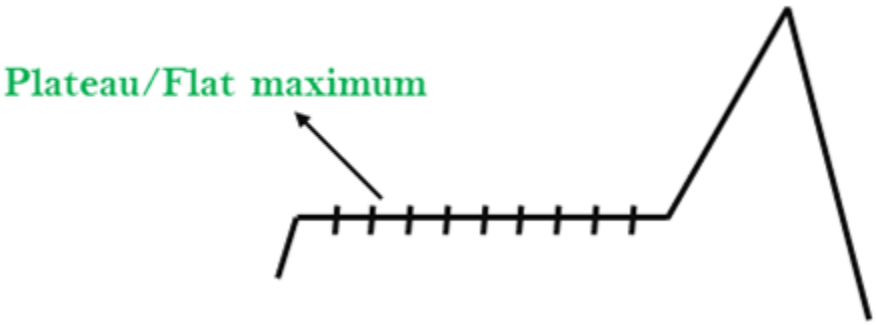
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



2. Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.

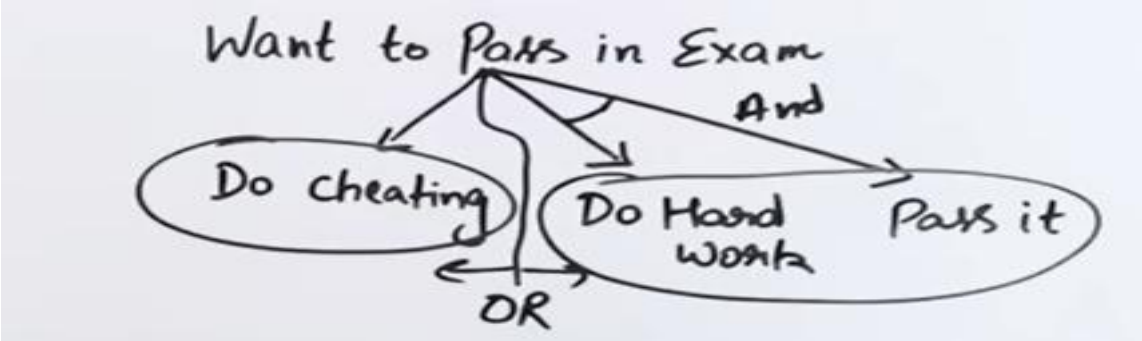


3. Adversarial Search: Used in competitive environments where agents (players) compete against each other, such as in board games like Chess or Go.

AO* SEARCH ALGORITHM

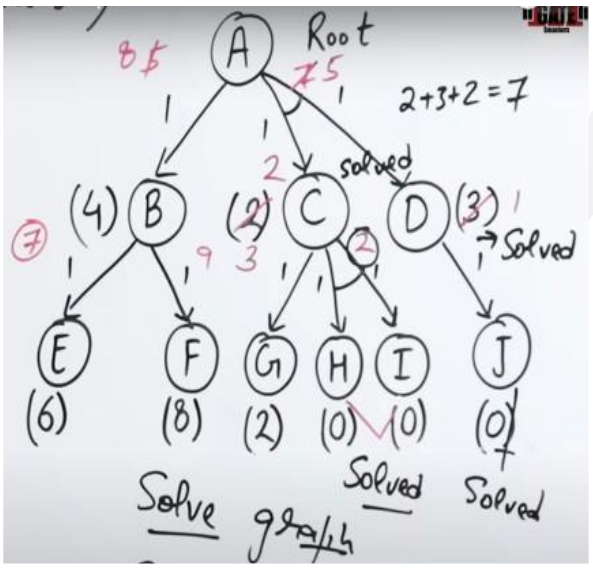
Problem Reduction In Artificial Intelligence

- AO* is informed search algorithm ,work based on heuristic
- We already know about the divide and conquer strategy, a solution to a problem can be obtained by decomposing it into smaller sub-problems.
- Each of this sub-problem can then be solved to get its sub solution.
- These sub solutions can then recombined to get a solution as a whole. That is called is Problem Reduction. AND-OR graphs or AND - OR trees are used for representing the solution.
- This method generates arc which is called as AND-OR arcs. One AND arc may point to any number of successor nodes, all of which must be solved in order for an arc to point to a solution.
- AND-OR graph is used to represent various kind of complex problem solutions.
- AO* search algo. is based on AND-OR graph so ,it is called AO* search algo.

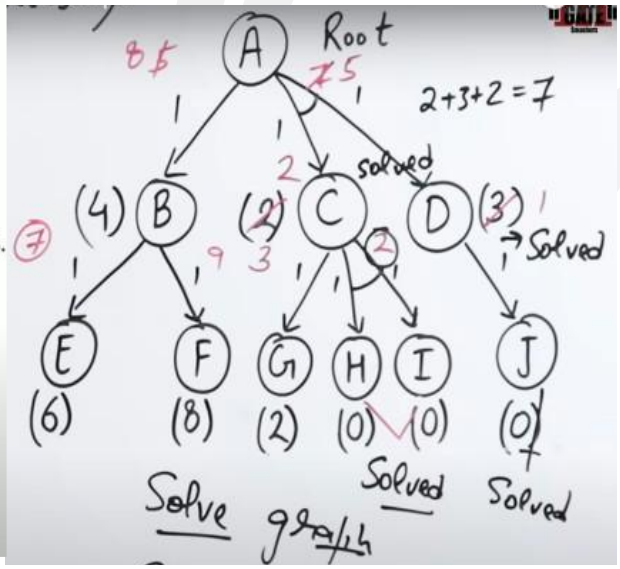


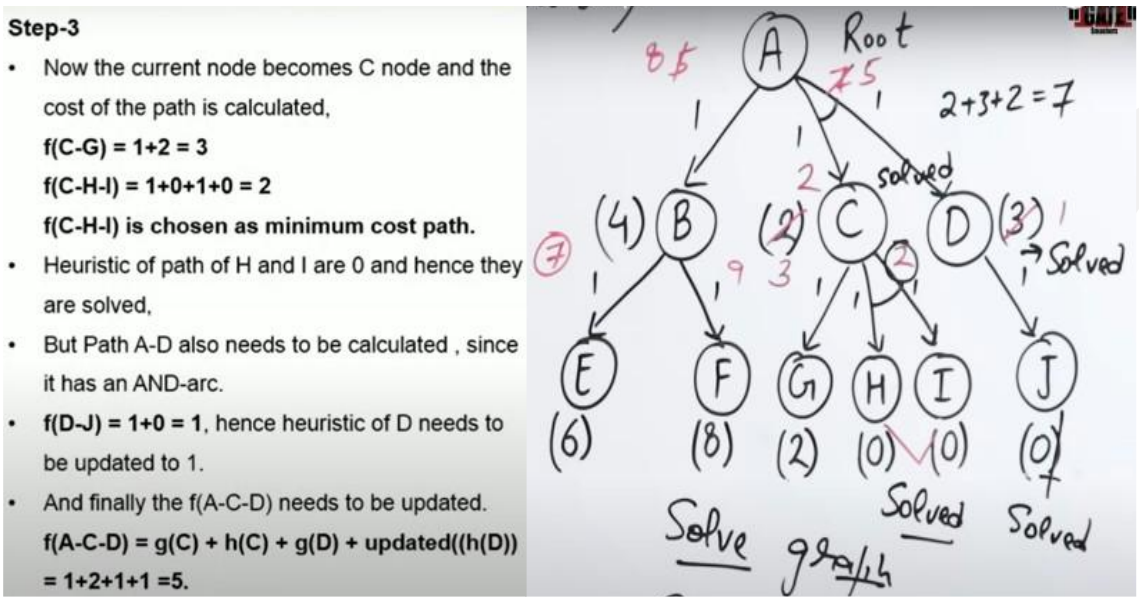
Step-1

- Starting from node A, we first calculate the best path.
- **f(A-B) = g(B) + h(B) = 1+4= 5** , where 1 is the default cost value of travelling from A to B and 4 is the estimated cost from B to Goal state.
- **f(A-C-D) = g(C) + h(C) + g(D) + h(D) = 1+2+1+3 = 7**, here we are calculating the path cost as both C and D because they have the AND-Arc.



- From the B node,
  f(B-E) = 1 + 6 = 7
  f(B-F) = 1 + 8 = 9
- Hence, the B-E path has lesser cost. Now the heuristics have to be updated since there is a difference between actual and heuristic value of B.
- The minimum cost path is chosen and is updated as the heuristic , in our case the value is 7.
- And because of change in heuristic of B there is also change in heuristic of A which is to be calculated again.
  f(A-B) = g(B) + updated((h(B)) = 1+7=8





2.3 Heuristic Algorithm vs. Solution-Guaranteed Algorithm

Heuristic Algorithms: These algorithms use approximate methods to find solutions quickly. They may not always guarantee the best or optimal solution but work well for large and complex problems. Example: Greedy Best-First Search, which selects the node that appears closest to the goal.

Solution-Guaranteed Algorithms: These algorithms guarantee finding a solution if one exists. They systematically explore all possible paths to ensure correctness. Example: Breadth-First Search (BFS) and A* Algorithm, which always find the shortest path if a solution exists.

> **Heuristics function:** Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Euclidean distance: $h(n)=\sqrt{(x_{Goal} - x_n)^2 + (y_{Goal} - y_n)^2}$

Manhattan distance: $h(n)=|x_{Goal} - x_n| + |y_{Goal} - y_n|$

2.4 Local Search and Optimization Problems

Local search algorithms operate by continuously improving a single solution rather than searching through an entire space. They are useful for optimization problems.

Hill Climbing:

- Hill Climbing is the simplest way to implement the heuristic search.
- The current state gets expanded and its child are evaluated at initial phase of Hill Climbing.
- In next phase, the best child may gets expansion, but this algorithms discards other information.
- Therefore, this algorithm halts when it realises there could be better path if sibling of current node was chosen by the algorithm.
- This is main drawback of Hill Climbing.
- This tendency to stuck over the Local Maxima is inspired from the ***blind mountain climber***.
- But, There is an algorithm called ***Best First Search*** which has the solution for this *Local Maxima*.

Best First Search

- Unlike the Hill Climbing, Best First search maintains/retains the states.(Without Discarding)
- For this, the Best First search Maintains two list : Open={ } and Closed={ }
- The Open List is a *priority queue*.(i.e maintained as sorted list)
- Initially, The root node placed on the open list.
- At each iteration, this algorithm picks the 1st element from the open list and checks equality with Goal State.
- If the 1st element of the open list is not equivalent to the Goal State, then algorithm march forward to *evaluate* all the descendants of that 1st element.
- Duplicate states are discarded from the open list and once an element *evaluated* it will be transferred to closed List.

A* Algorithm: Uses both cost and heuristic values to find the best solution. Guarantees an optimal solution if the heuristic is admissible (i.e., it never overestimates the actual cost).

AO* Algorithm (AND-OR Graphs): Works in decision-making problems where multiple solutions can contribute to the goal. Useful in AI planning and expert systems.

2.5 Adversarial Search

Adversarial search is used in AI when there are competing agents (opponents). This occurs in strategic decision-making, such as in Chess or Tic-Tac-Toe.

Minimax Algorithm: Assumes that the opponent plays optimally. The AI selects moves that minimize the opponent's maximum possible gain.

Alpha-Beta Pruning: Optimizes Minimax by cutting off branches that do not influence the final decision. Reduces the number of nodes that need to be evaluated, making it more efficient.

Monte Carlo Tree Search (MCTS): Used in games like Go to simulate possible moves and improve decision-making. Balances exploration (trying new moves) and exploitation (choosing the best-known move).

2.6 AI and Game Playing

Game-playing AI uses search techniques and heuristics to make intelligent decisions.

Examples of AI in Games: Chess-playing AI: Uses Minimax with heuristics to evaluate board positions. AlphaGo: Uses deep learning and Monte Carlo Tree Search to master the game of Go. Reinforcement Learning (RL): AI learns by trial and error through rewards and penalties.

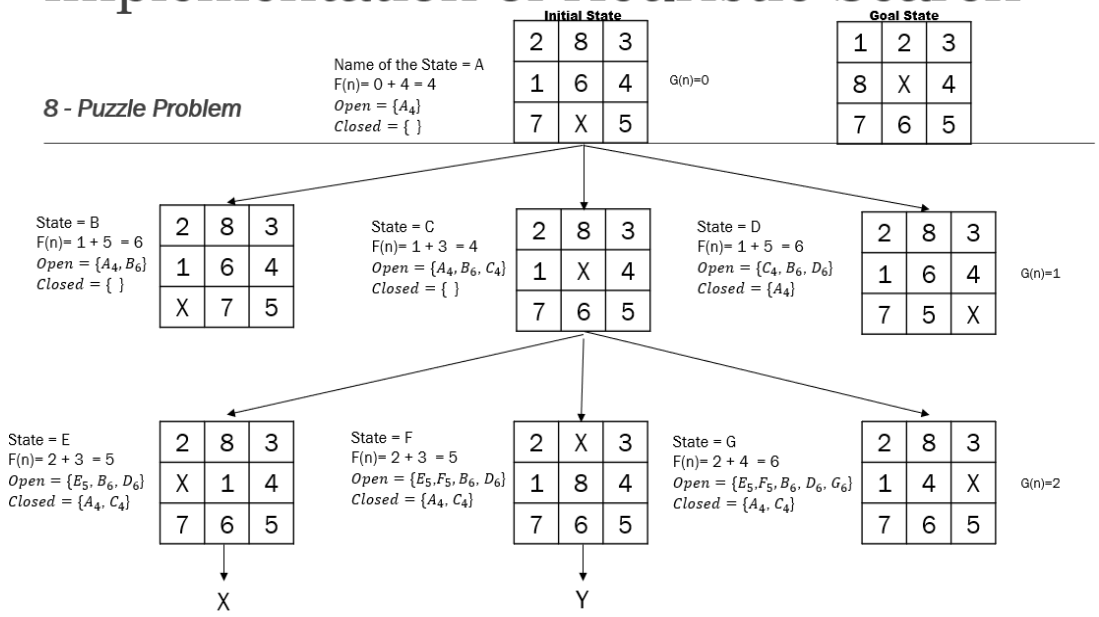
Key Techniques in AI Game Playing:

- State Representation: Defines the current game status (e.g., board configuration in Chess).

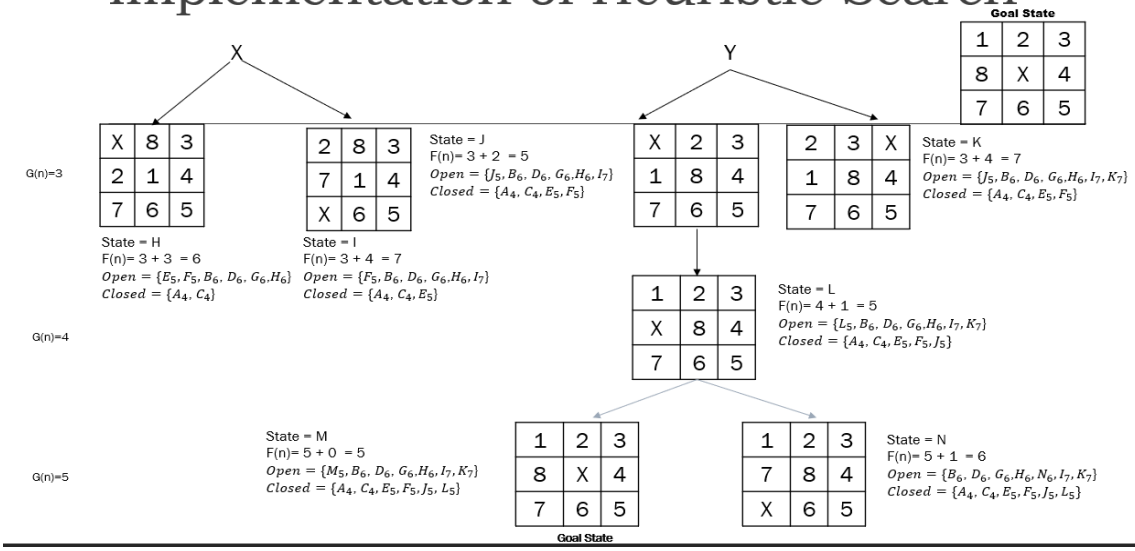
- Search Algorithms: Helps in decision-making (e.g., BFS, DFS, A*).

- Machine Learning: Enables AI to improve through experience (e.g., AlphaZero learning Chess).

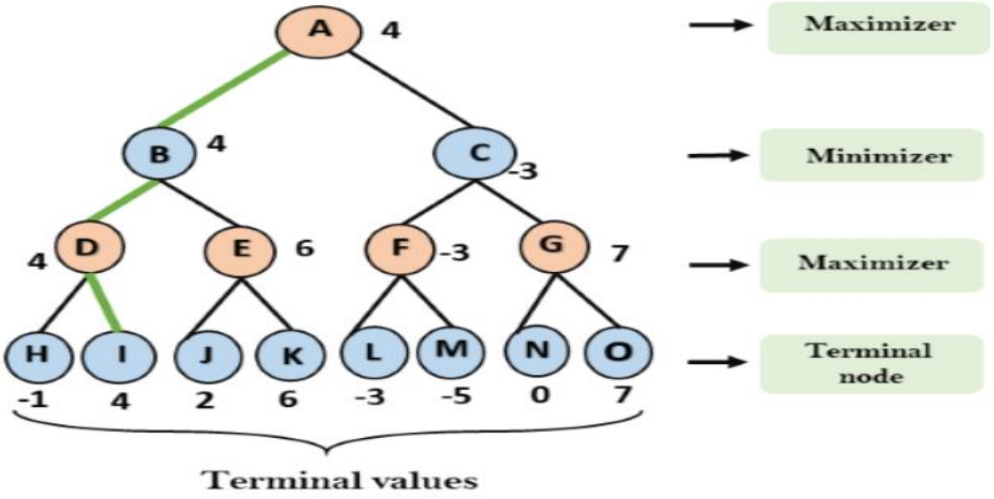
Implementation of Heuristic Search



Implementation of Heuristic Search



MIN-MAX Game Theory Example



- **Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.**
- **Mini-Max algorithm uses recursion to search through the game-tree.**
- **Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.**

UNIT-3

Knowledge Representation:

Knowledge Representation: Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge, they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**.

3.1 What to Represent?

In AI, knowledge encompasses the information and data that an intelligent system must process to perform tasks effectively. This includes facts about the world, relationships between entities, and heuristics for problem-solving.

- **Object:** All the facts about objects in our world domain. E.g., Guitars contain strings, and trumpets are brass instruments.
- **Events:** Events are the actions that occur in our world.
- **Performance:** It describes behavior involving knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge: Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarativesentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta-knowledge:

- Knowledge about the other types of knowledge is called Meta-knowledge.

4. Heuristic knowledge:

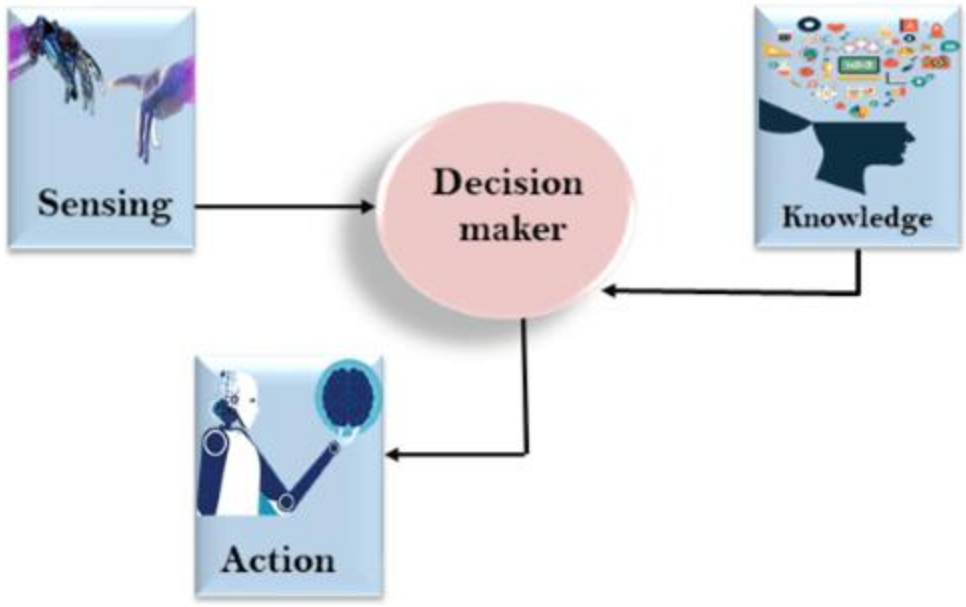
- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

5. Structural knowledge:

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

The relation between knowledge and intelligence:

Knowledge of the real world plays a vital role in intelligence and the same for creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input. Let's suppose you meet some person who is speaking in a language that you don't know, then how you will be able to act on that? The same thing applies to the intelligent behavior of the agents. As we can see in the below diagram, there is one decision-maker who acts by sensing the environment and using knowledge. But if the knowledge part is not present then, it cannot display intelligent behavior.



3.2 Properties of Knowledge Representation System:

A good knowledge representation system must possess the following properties.

1. **1. Representational Accuracy:**
   KR system should have the ability to represent all kind of required knowledge.
2. **2. Inferential Adequacy:**
   KR system should have the ability to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
3. **3. Inferential Efficiency:**
   The ability to direct the inferential knowledge mechanism in the most productive directions by storing appropriate guides.
4. **4. Acquisitional efficiency-** The ability to acquire new knowledge easily using automatic methods.

Approaches:

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 of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

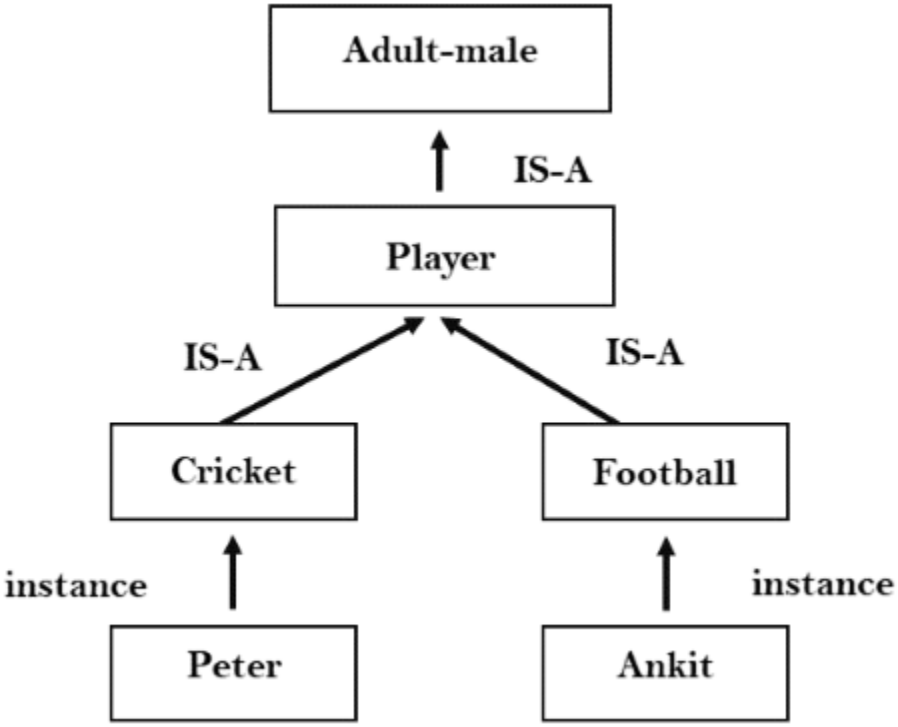
Example: The following is the simple relational knowledge representation.

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.

- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.
- **Example:**



3. Inferential knowledge:

- The inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guarantees correctness.
- **Example:** Let's suppose there are two statements:
    - Marcus is a man
    - All men are mortal
      Then it can be represented as;

      man(Marcus)
      ∀x = man (x) ----------> mortal (x)s

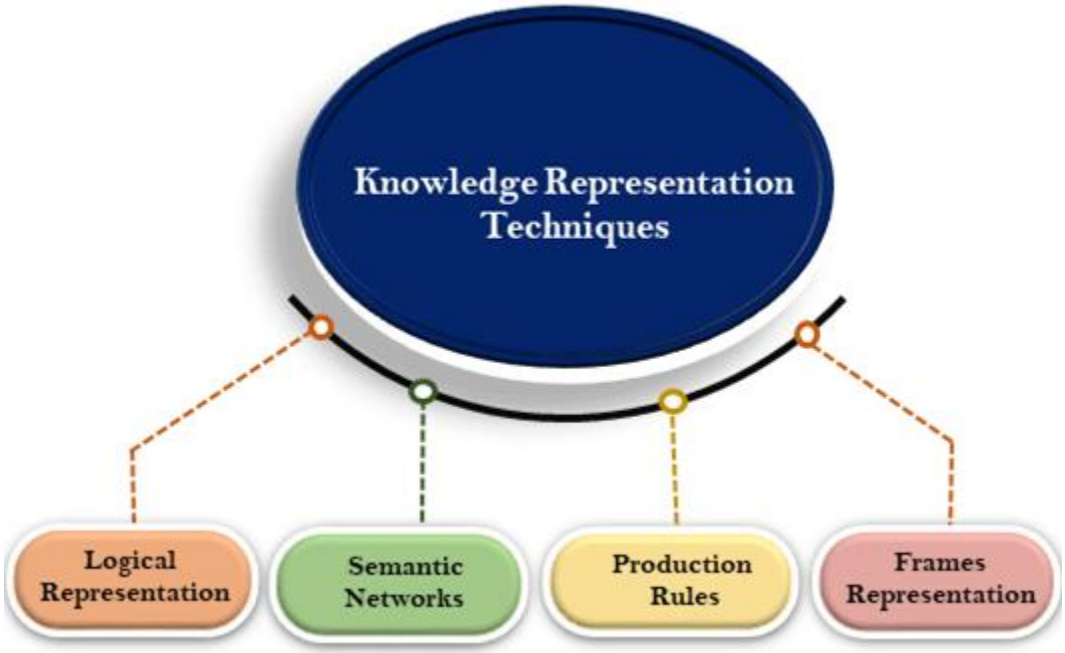
4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describe how to do specific things, and how to proceed.
- In this approach, one important rule is used which is the **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 that we can represent all cases in this approach.

3.3 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 some concrete rules which deals with propositions and has no 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 using syntax and semantics.

Syntax:

- Syntaxes 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 categorised into mainly two logics:

1. Propositional Logics
2. Predicate logics

Advantages of logical representation:

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

Disadvantages of logical Representation:

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.

Note: Do not be confused with logical representation and logical reasoning as logical representation is a representation language and reasoning is a process of thinking logically.

2. Semantic Network Representation

Semantic networks are 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 are easy to understand and can be easily extended.This representation consist of mainly two types of relations:

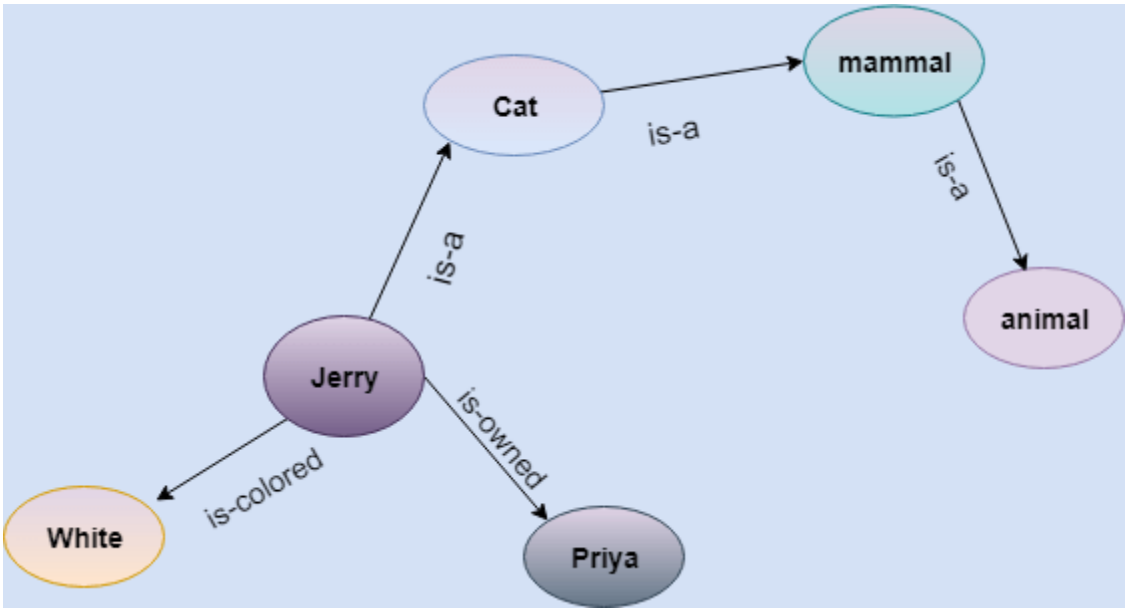
1. IS-A relation (Inheritance)
2. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.

4. Jerry is brown-colored.
5. All Mammals are animal.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

3. Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

Facets: The various aspects of a slot is known as **Facets**. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example: 1

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

Slots	Filters
Title	**Artificial Intelligence**
Genre	**Computer Science**
Author	**Peter Norvig**
Edition	**Third Edition**
Year	**1996**
Page	**1152**

Example 2:

Let's 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	Filter
Name	**Peter**
Profession	**Doctor**
Age	**25**
Marital status	**Single**

Weight	78

Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

4. Production Rules

Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:

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

In production rules agent checks for the condition 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. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.

If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

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:

1. The production rules are expressed in natural language.

2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

Disadvantages of Production rule:

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. During the execution of the program, many rules may be active hence rule-based production systems are inefficient.

Propositional logic in Artificial intelligence (out of syllabus Extra)

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.

Example:

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)
4. d) 5 is a prime number.

Syntax of propositional logic:

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

1. **Atomic Propositions**
2. **Compound propositions**

- **Atomic Proposition:** 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:

1. a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.
2. b) "The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

1. a) "It is raining today, and street is wet."
2. b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has ∧ connective such as, **P ∧ Q** is called a conjunction.
   Example: Rohan is intelligent and hardworking. It can be written as,
   P= Rohan is intelligent,
   Q= Rohan is hardworking. → P∧ Q.
3. **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P ∨ Q**.
4. **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as
   If it is raining, then the street is wet.
   Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
5. **Biconditional:** A sentence such as **P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive**
   P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.

Following is the summarized table for Propositional Logic Connectives:

Connective symbols	Word	Technical term	Example
∧	AND	Conjunction	A ∧ B
∨	OR	Disjunction	A ∨ B
→	Implies	Implication	A → B
⇔	If and only if	Biconditional	A⇔ B
¬ or ~	Not	Negation	¬ A or ¬ B

Truth Table:

In propositional logic, we need to know the truth values of propositions in all possible scenarios. 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 table for all logical connectives:

For Negation:

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**

Truth table with three propositions:

We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

P	Q	R	¬R	Pv Q	PvQ→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

3.4 Reasoning in Artificial intelligence

In previous topics, we have learned various ways of knowledge representation in artificial intelligence. Now we will learn the various ways to reason on this knowledge using different logical schemes.

Reasoning:

The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or we can say, "**Reasoning is a way to infer facts from existing data**." It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

Types of Reasoning

In artificial intelligence, reasoning can be divided into the following categories:

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning
- Common Sense Reasoning
- Monotonic Reasoning
- Non-monotonic Reasoning

Note: Inductive and deductive reasoning are the forms of propositional logic.

1. Deductive reasoning:

Deductive reasoning is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.

Deductive reasoning is a type of propositional logic in AI, and it requires various rules and facts. It is sometimes referred to as top-down reasoning, and contradictory to inductive reasoning.

In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.

Deductive reasoning mostly starts from the general premises to the specific conclusion, which can be explained as below example.

Example:

Premise-1: All the human eats veggies

Premise-2: Suresh is human.

Conclusion: Suresh eats veggies.

The general process of deductive reasoning is given below:

2. Inductive Reasoning:

Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with the series of specific facts or data and reaches to a general statement or conclusion.

Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.

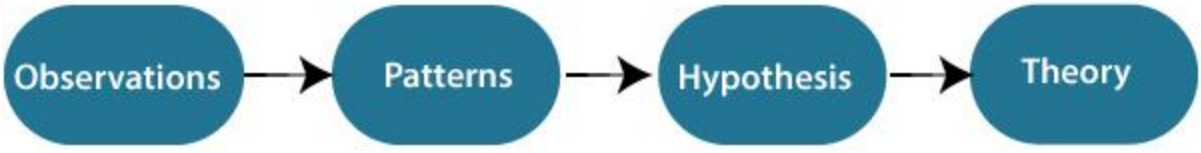
In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.

In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.

Example:

Premise: All of the pigeons we have seen in the zoo are white.

Conclusion: Therefore, we can expect all the pigeons to be white.



3. Abductive reasoning:

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.

Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, the premises do not guarantee the conclusion.

Example:

Implication: Cricket ground is wet if it is raining

Axiom: Cricket ground is wet.

Conclusion It is raining.

4. Common Sense Reasoning

Common sense reasoning is an informal form of reasoning, which can be gained through experiences.

Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day.

It relies on good judgment rather than exact logic and operates on **heuristic knowledge** and **heuristic rules**.

Example:

1. **One person can be at one place at a time.**
2. **If I put my hand in a fire, then it will burn.**

The above two statements are the examples of common sense reasoning which a human mind can easily understand and assume.

5. Monotonic Reasoning:

In 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. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Any theorem proving is an example of monotonic reasoning.

Example:

- **Earth revolves around the Sun.**

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

Advantages of Monotonic Reasoning:

- In monotonic reasoning, each old proof will always remain valid.
- If we deduce some facts from available facts, then it will remain valid for always.

Disadvantages of Monotonic Reasoning:

- We cannot represent the real world scenarios using Monotonic reasoning.
- Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
- Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

6. Non-monotonic Reasoning

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example: Let suppose the knowledge base contains the following knowledge:

- **Birds can fly**
- **Penguins cannot fly**
- **Pitty is a bird**

So from the above sentences, we can conclude that **Pitty can fly**.

However, if we add another sentence into the knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

Advantages of Non-monotonic reasoning:

- For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
- In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

Disadvantages of Non-monotonic Reasoning:

- In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
- It cannot be used for theorem **proving**.

Unit-4

4.1 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence (AI) that enables systems to learn and improve from experience without being explicitly programmed. It involves developing algorithms that can analyze data, identify patterns, and make decisions with minimal human intervention.

Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Types of Machine Learning:

1. **Supervised Learning -** The model learns from labeled data.

2. Unsupervised Learning - The model learns patterns from unlabeled data.

3. Semi-supervised learning: This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

4. Reinforcement Learning - The model learns by interacting with the environment and receiving rewards or penalties.

4.2 Statistical or Unsupervised Learning

Unsupervised learning is a type of ML where the model finds patterns in data without explicit labels. It is commonly used for clustering and association tasks.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

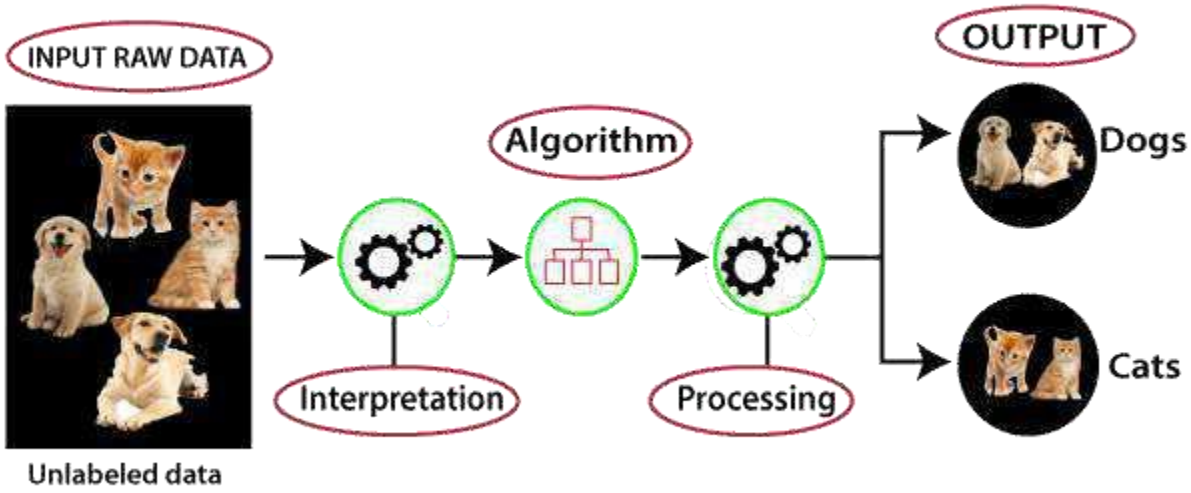
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:



Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc. Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Techniques in Unsupervised Learning:

There are mainly 3 types of Algorithms which are used for Unsupervised dataset.

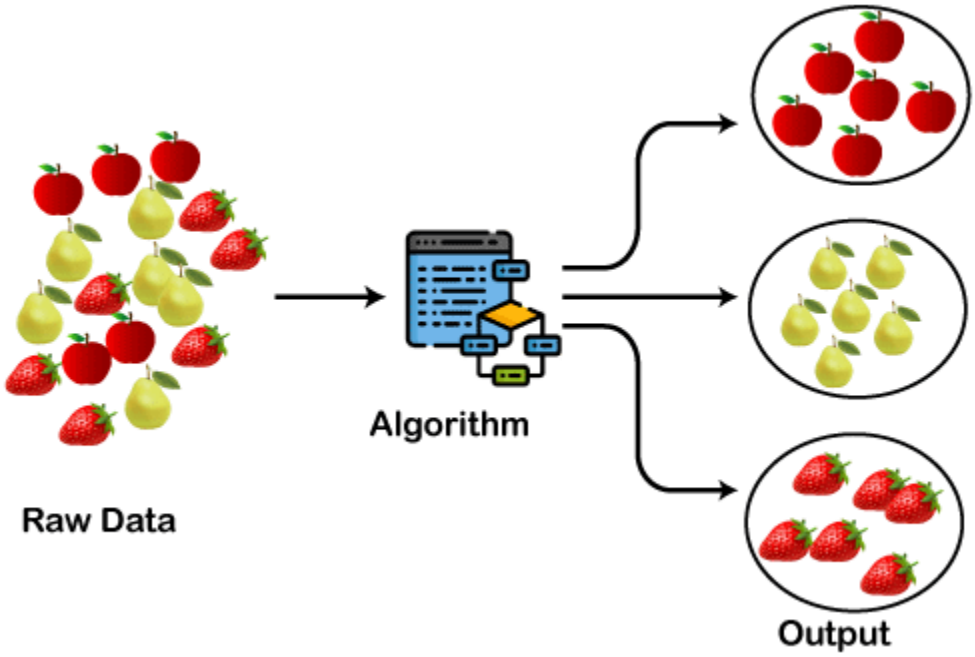
- **Clustering**
- **Association Rule Learning**
- **Dimensionality Reduction**

Clustering in unsupervised machine learning is the process of grouping unlabeled data into clusters based on their similarities. The goal of clustering is to identify patterns and relationships in the data without any prior knowledge of the data's meaning.

Some common clustering algorithms:

- ***K-means Clustering:*** *Groups data into K clusters based on how close the points are to each other.*
- ***Hierarchical Clustering:*** *Creates clusters by building a tree step-by-step, either merging or splitting groups.*
- ***Density-Based Clustering (DBSCAN):*** *Finds clusters in dense areas and treats scattered points as noise.*

The below diagram explains the workings of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



2. Association Rule Learning

Association rule learning is also known as association rule mining is a common technique used to discover associations in unsupervised machine learning. This technique is a rule-based ML technique that finds out some very useful relations between parameters of a large data set. This technique is basically used for market basket analysis that helps to better understand the relationship between different products.

Some common Association Rule Learning algorithms:

- ***Apriori Algorithm:*** *Finds patterns by exploring frequent item combinations step-by-step.*
- ***FP-Growth Algorithm:*** *An Efficient Alternative to Apriori. It quickly identifies frequent patterns without generating candidate sets.*
- ***Eclat Algorithm:*** *Uses intersections of itemsets to efficiently find frequent patterns.*
- ***Efficient Tree-based Algorithms:*** *Scales to handle large datasets by organizing data in tree structures.*

3. Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features in a dataset while preserving as much information as possible. This technique is useful for improving the performance of machine learning algorithms and for data visualization.

Here are some popular
Dimensionality Reduction algorithms*:*

- ***Principal Component Analysis (PCA)**: Reduces dimensions by transforming data into uncorrelated principal components.*
- ***Linear Discriminant Analysis (LDA)**: Reduces dimensions while maximizing class separability for classification tasks.*

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

4.3 ML Properties

1. Generalization

- Generalization refers to the model's ability to perform well on unseen or new data.
- A well-generalized model can apply what it has learned from training data to real-world scenarios.
- Poor generalization can lead to overfitting or underfitting.

2. Overfitting

- Overfitting occurs when a model learns the noise or random fluctuations in the training data rather than the actual pattern.
- This results in high accuracy on training data but poor performance on new, unseen data.
- Techniques to prevent overfitting include:
  - Cross-validation

    - Regularization (L1, L2)
    - Pruning (for decision trees)
    - Using simpler models

3. Underfitting

- Underfitting happens when a model is too simple and fails to capture the underlying patterns in the data.
- This leads to poor performance on both training and test datasets.
- Solutions include:
    - Using more complex models
    - Increasing training time
    - Adding more relevant features

4. Bias-Variance Tradeoff

- **Bias**: The error due to overly simplistic models that do not capture complexity well. High bias leads to underfitting.
- **Variance**: The error due to overly complex models that capture noise along with the pattern. High variance leads to overfitting.
- The goal is to find a balance between bias and variance to achieve optimal model performance.

5. Scalability

- The ability of a model to handle large datasets efficiently.
- Some models (e.g., deep learning) scale well with big data, while others (e.g., k-NN) may struggle.
- Methods to improve scalability:
    - Using parallel processing
    - Implementing distributed computing
    - Feature selection to reduce dimensionality

6. Interpretability

- Some models, such as linear regression and decision trees, are easy to interpret.
- Complex models like deep learning and ensemble methods (e.g., random forests, gradient boosting) offer high accuracy but are often difficult to interpret.
- Interpretability is crucial in applications like healthcare, finance, and legal decision-making.

7. Robustness

- A robust model performs well even when given slightly altered or noisy input data.
- It should not be overly sensitive to small changes in data distribution.

- Techniques to enhance robustness:
    - Data augmentation
    - Outlier detection
    - Regularization

4.4 Reinforcement Learning

Reinforcement Learning (RL) is an area of ML where an agent learns to make decisions by taking actions in an environment to maximize cumulative rewards.

Here are some important terms used in Reinforcement AI:

1. **Agent** - The entity that makes decisions.
2. **Environment** - The setting in which the agent operates.
3. **State** - A representation of the current situation of the agent.
4. **Action** - A move the agent can make.
5. **Reward** - Feedback received after taking an action.
6. **Policy** - A strategy the agent follows to decide actions.
7. **Q-Learning** - A value-based reinforcement learning algorithm.
8. **Deep Q-Networks (DQN)** - A combination of deep learning and reinforcement learning.

Applications:

- Robotics
- Game playing (e.g., AlphaGo)
- Autonomous vehicles
- Personalized recommendations

How Reinforcement Learning works?

Let's see some simple example which helps you to illustrate the reinforcement learning mechanism.

Consider the scenario of teaching new tricks to your cat

- As cat doesn't understand English or any other human language, we can't tell her directly what to do. Instead, we follow a different strategy.
- We emulate a situation, and the cat tries to respond in many different ways. If the cat's response is the desired way, we will give her fish.
- Now whenever the cat is exposed to the same situation, the cat executes a similar action with even more enthusiastically in expectation of getting more reward(food).

- That's like learning that cat gets from "what to do" from positive experiences.
- At the same time, the cat also learns what not do when faced with negative experiences.

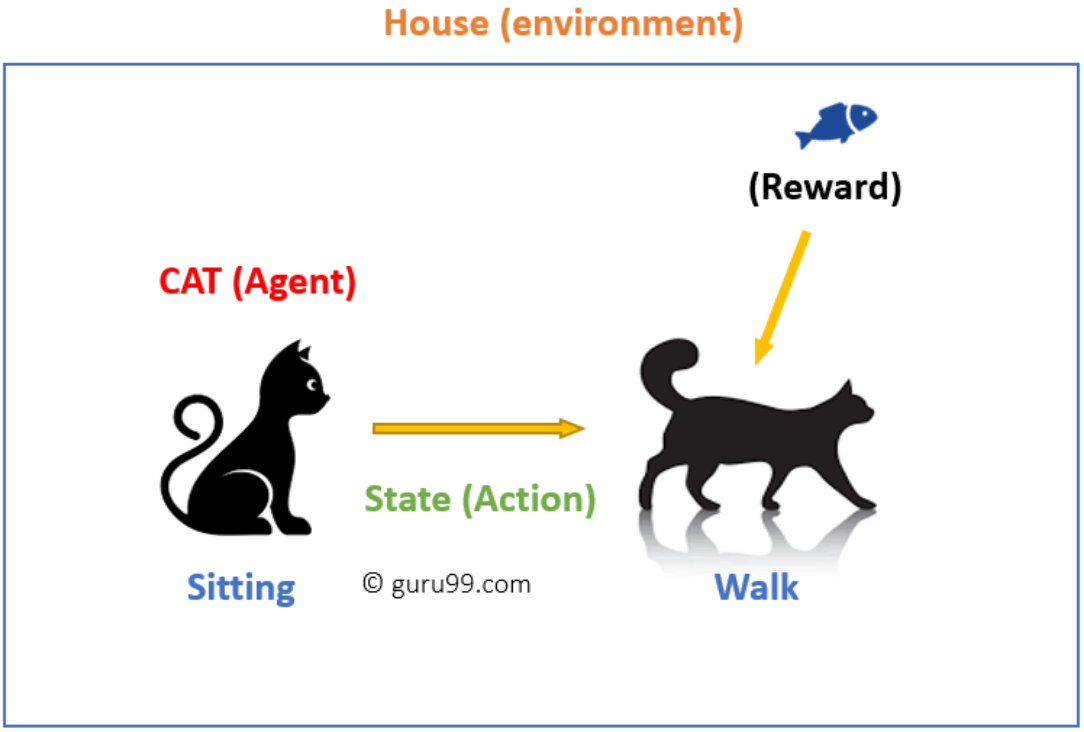
Main points in Reinforcement learning –

- Input: The input should be an initial state from which the model will start
- Output: There are many possible outputs as there are a variety of solutions to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

Types of Reinforcement: There are two types of Reinforcement:

1. **Positive –**
   Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.
   Advantages of reinforcement learning are:
   - Maximizes Performance
   - Sustain Change for a long period of time
   - Too much Reinforcement can lead to an overload of states which can diminish the results
2. **Negative –**
   Negative Reinforcement is defined as strengthening of behavior because a negative condition is stopped or avoided.
   Advantages of reinforcement learning:
   - Increases Behavior
   - Provide defiance to a minimum standard of performance
   - It Only provides enough to meet up the minimum behavior

Example of Reinforcement Learning



4.5 Decision Tree

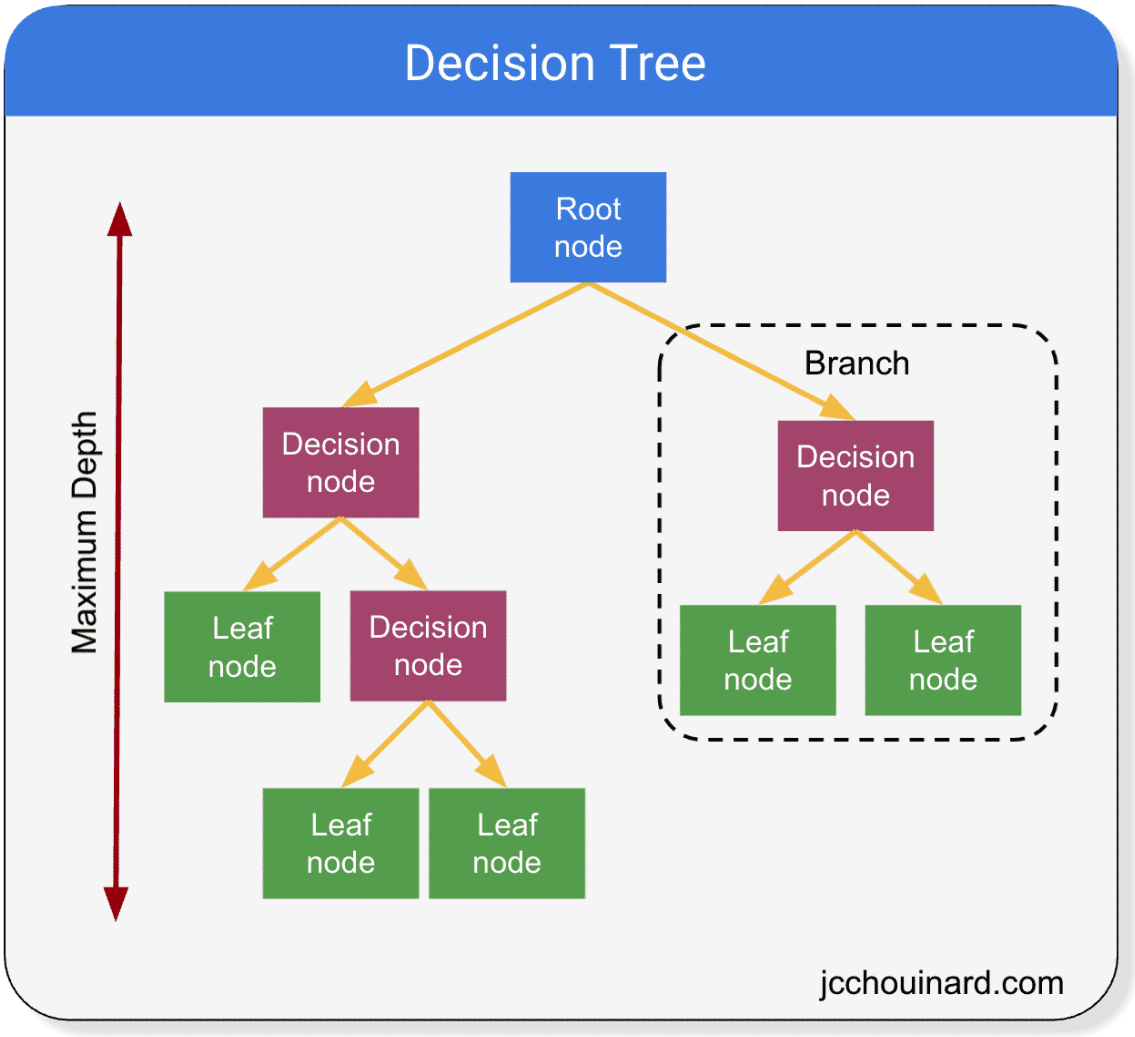
A Decision Tree is a supervised learning algorithm used for classification and regression tasks. It splits the data into branches based on feature values to reach a decision.

Components of a Decision Tree:

1. Root Node - The starting point containing the entire dataset.

2. Internal Nodes - Decision points splitting the data.

3. Leaf Nodes - Terminal nodes that represent outcomes.



Decision Tree Terminologies

- **Root Node:** The root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

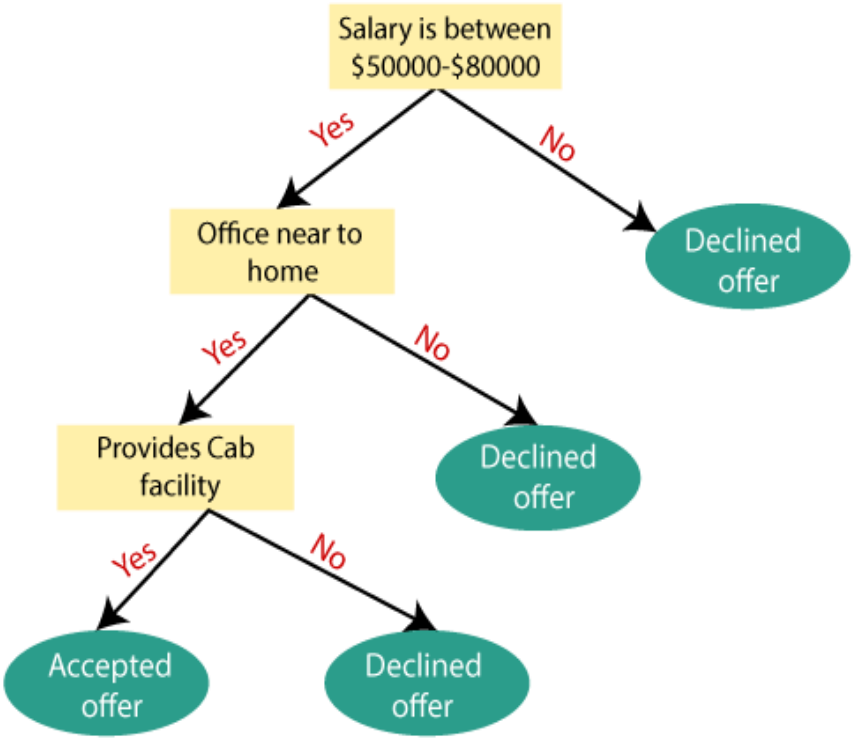
How does the Decision Tree algorithm Work?

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm

compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

- For the next node, the algorithm again compares the attribute value with the other subnodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:
- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer).
Consider the below diagram:



Advantages:

- Easy to interpret and visualize.

- Handles both numerical and categorical data.

- Requires little data preprocessing.

Disadvantages:

- Prone to overfitting if not pruned properly.

- Can be biased towards dominant features.

Applications:

- Medical diagnosis

- Credit scoring

- Fraud detection

- Customer segmentation

UNIT-5

Pattern Recognition

5.1 Introduction to Pattern Recognition

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms.
Example: The colors on the clothes, speech pattern, etc. In computer science, a pattern is represented using vector feature values.Pattern recognition is **the process of recognizing patterns by using a machine learning algorithm**. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of pattern recognition is its application potential.

Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis. In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves the classification and cluster of patterns.

Advantages:

- Pattern recognition solves classification problems
- Pattern recognition solves the problem of fake biometric detection.
- It is useful for cloth pattern recognition for visually impaired blind people.
- It helps in speaker diarization.
- We can recognize particular objects from different angles.

Disadvantages:

- The syntactic pattern recognition approach is complex to implement, and it is a very slow process.
- Sometimes, to get better accuracy, a larger dataset is required.
- It cannot explain why a particular object is recognized.

Example: my face vs my friend's face.

Applications of Pattern Recognition

- Biometric Recognition – Face, fingerprint, and iris recognition.

• Medical Diagnosis – Detecting diseases from medical images.

• Speech Recognition – Converting spoken words into text.

• Handwriting Recognition – Used in digital notepads and banking check processing.

• Robotics & Autonomous Systems – Identifying objects and navigation.

Detailed applications:

- **Image processing, segmentation, and analysis**

  Pattern recognition is used to give human recognition intelligence to machines that are required in image processing.
- **Computer vision**

  Pattern recognition is used to extract meaningful features from given image/video samples and is used in computer vision for various applications like biological and biomedical imaging.
- **Seismic analysis**

  The pattern recognition approach is used for the discovery, imaging, and interpretation of temporal patterns in seismic array recordings. Statistical pattern recognition is implemented and used in different types of seismic analysis models.
- **Radar signal classification/analysis**

  Pattern recognition and signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.
- **Speech recognition**

  The greatest success in speech recognition has been obtained using pattern recognition paradigms. It is used in various algorithms of speech recognition which tries to avoid the problems of using a phoneme level of description and treats larger units such as words as pattern
- **Fingerprint identification**

  Fingerprint recognition technology is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches are widely used.

5.2 Design Principles of a Pattern Recognition System

Data Collection

Gather raw data (e.g., images, audio, or sensor data).

Preprocessing

Enhancing data quality by removing noise, normalization, and feature extraction.

Feature Extraction

Selecting meaningful characteristics from data to improve classification.

Classification / Clustering

Supervised Learning (classification) and Unsupervised Learning (clustering).

Post-processing & Decision Making

Final stage where decisions are made based on recognized patterns.

5.3 Statistical Pattern Recognition System

Statistical Approach:

Statistical methods are mathematical formulas, models, and techniques that are used in the statistical analysis of raw research data. The application of statistical methods extracts information from research data and provides different ways to assess the robustness of research outputs.

Two main statistical methods are used :

1.Descriptive Statistics: It summarizes data from a sample using indexes such as the mean or standard deviation.

2.Inferential Statistics: It draw conclusions from data that are subject to random variation. **Structural Approach:**

The Structural Approach is a technique wherein the learner masters the pattern of sentence. Structures are the different arrangements of words in one accepted style or the other.

Types of structures:

- Sentence Patterns
- Phrase Patterns
- Formulas
- Idioms

Difference Between Statistical Approach and Structural Approach:

Sr. No.	Statistical Approach	Structural Approach
1	Statistical decision theory.	Human perception and cognition.
2	Quantitative features.	Morphological primitives
3	Fixed number of features.	Variable number of primitives.
4	Ignores feature relationships.	Captures primitives relationships.
5	Semantics from feature position.	Semantics from primitives encoding.
6	Statistical classifiers.	Syntactic grammars.

Statistical pattern recognition uses probability and statistical techniques to classify data.

Key Concepts:

• Bayes' Theorem – Used for probabilistic classification.

• Decision Boundaries – Separating different classes based on statistical properties.

• Nearest Neighbor Classifier – Classifies data based on similarity to known examples.

5.4 Machine Perception

Machine perception allows computers to interpret sensory information like images, sound, and touch. Machine Perception refers to the added functionality in computer systems that enables reaction based on senses, similar to human perception. Computers now have the added capacity to see, hear, touch, and in some cases even smell

What are the business advantages of using Machine Perception?

- Predictive functionality: Accessing data that is processed through human-like senses is the closest alternative to consumer testing. Machine perception can help a business predict how a consumer or user will see, hear, and experience a new product, site, or service.

- Robotics: Machines with robotic capabilities are advancing the manufacturing and production sectors, but organizations can significantly reduce the number of malfunctions with the added capabilities of machine vision or tactile responsiveness. Smarter robots that can detect visible errors and respond to equipment failure can save the organization costly repairs and replacements.

- Accuracy: Collecting and analyzing data with computational methods is an exact science. Even analyzing through models based on human senses will be more accurate than human analysis alone.

- Efficiency and productivity: Computer analysis and computer processing are much faster than human employees can physically function. Reducing the number of errorprone tasks that are carried out by humans will reduce both errors and time spent.

- Recommendations: Machine perception empowers the use of predictive analytics, but aside from predicting customer reactions, businesses can also forecast what consumers will like and buy. This provides an additional opportunity for revenue by recommending additional products and services based on data-backed customer preferences.

Types of Machine Perception:

• Image Perception – Recognizing objects in images.

• Speech Perception – Converting speech into text.

• Tactile Perception – Robots sensing physical contact.

5.5 Line Finding and Intersection

Line detection helps in identifying boundaries, roads, and objects in images.

Techniques for Line Detection:

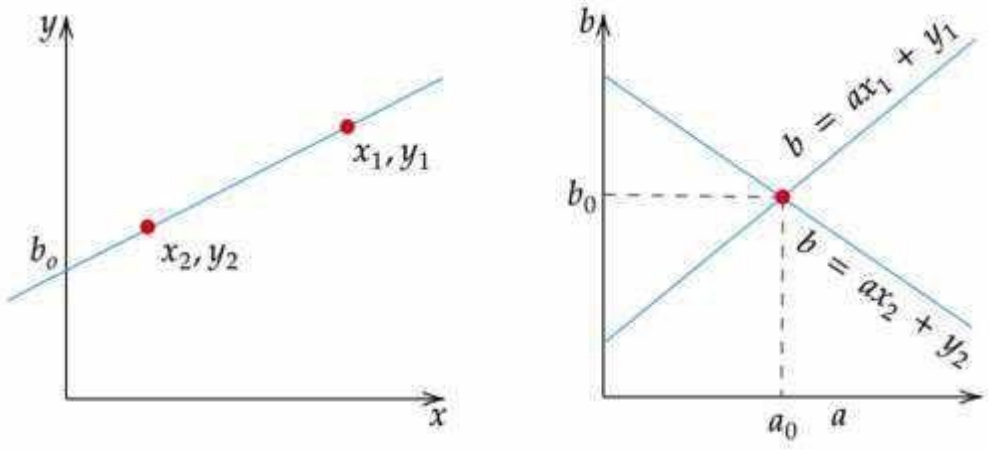
• Hough Transform – Detects straight lines in an image.

The Hough Transform is an algorithm patented by Paul V. C. Hough and was originally invented to recognize complex lines in photographs (Hough, 1962). Since its inception, the algorithm has been modified and enhanced to be able to recognize other shapes such as circles and quadrilaterals of specific types. In order to understand how the Hough

Transform algorithm works, it is important to understand four concepts: edge image, the Hough Space, and the mapping of edge points onto the Hough Space, an alternate way to represent a line, and how lines are detected.

• Edge Detection (Sobel, Canny) – Finds edges of objects.

The Hough Space and the Mapping of Edge Points onto the Hough Space



The Hough Space is a 2D plane that has a horizontal axis representing the slope and the vertical axis representing the intercept of a line on the edge image. A line on an edge image is represented in the form of $y = ax + b$ (Hough, 1962). One line on the edge image produces a point on the Hough Space since a line is characterized by its slope a and intercept b. On the other hand, an edge point (x_i, y_i) on the edge image can have an infinite number of lines pass through it.
Therefore, an edge point produces a line in the Hough Space in the form of $b = ax_i + y_i$ (Leavers, 1992). In the Hough Transform algorithm, the Hough Space is used to determine whether a line exists in the edge image.

An Alternate Way to Represent a Line

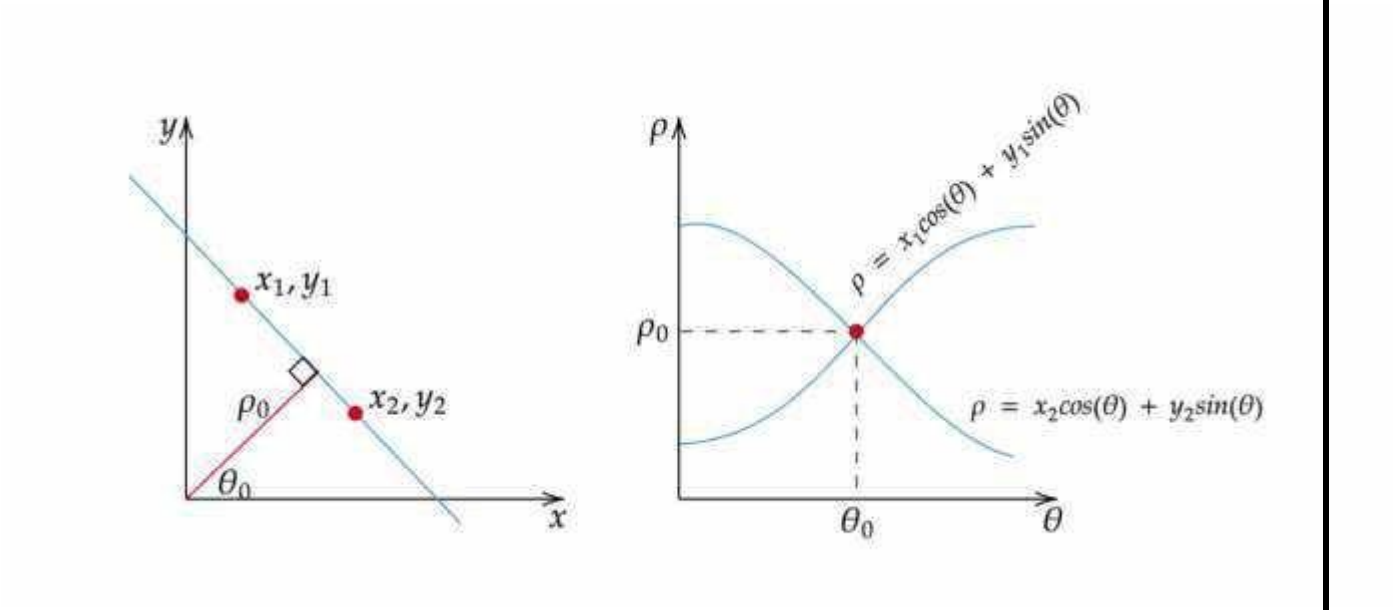
$$m = \frac{rise}{run} = \frac{y_2 - y_1}{x_2 - x_1}$$

m = slope

(x_1, y_1) = first point

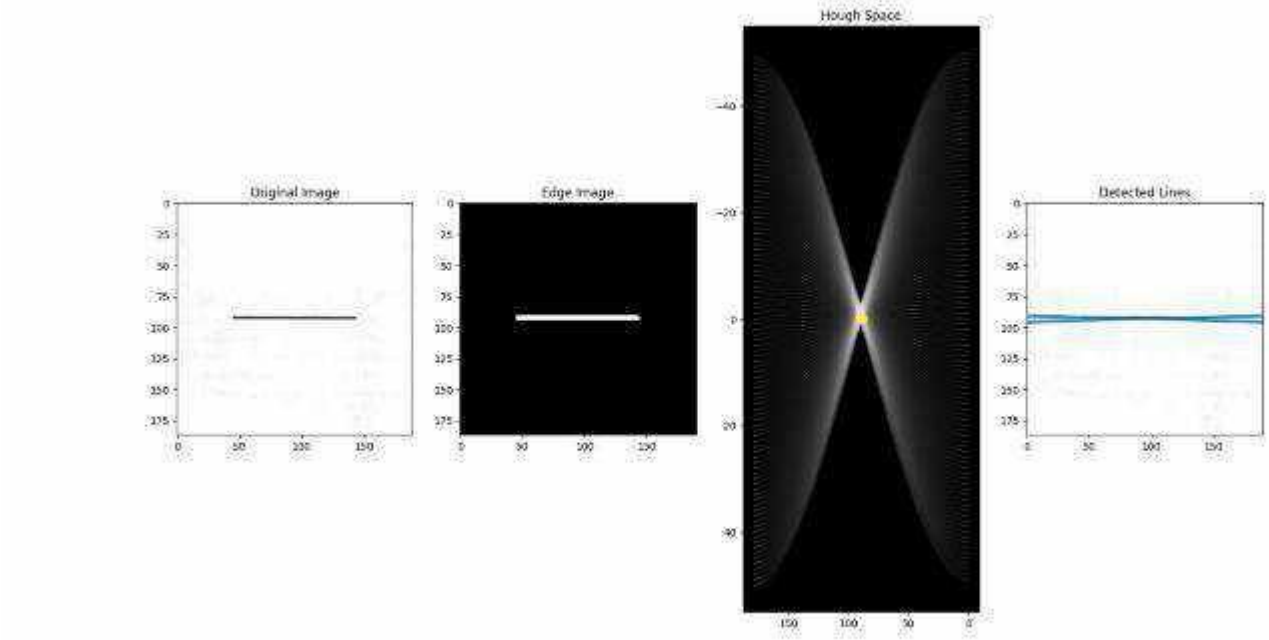
(x_2, y_2) = second point

There is one flaw with representing lines in the form of $y = ax + b$ and the Hough Space with the
slope and intercept. In this form, the algorithm won't be able to detect vertical lines because the slope a is undefined/infinity for vertical lines (Leavers, 1992). Programmatically, this means that a computer would need an infinite amount of memory to represent all possible values of a. To avoid this issue, a straight line is instead represented by a line called the normal line that passes through the origin and perpendicular to that straight line. The form of the normal line is $\rho = x\cos(\theta) + y\sin(\theta)$ where ρ is the length of the normal line and θ is the angle between the normal line and the x axis.



Using this, instead of representing the Hough Space with the slope *a* and intercept *b*, it is now represented with ρ and θ where the horizontal axis is for the θ values and the vertical axis are for the ρ values. The mapping of edge points onto the Hough Space works in a similar manner except that an edge point (x_i, y_i) now generates a cosine curve in the Hough Space instead of a straight line (Leavers, 1992). This normal representation of a line eliminates the issue of the unbounded value of *a* that arises when dealing with vertical lines.

Line Detection



As mentioned, an edge point produces a cosine curve in the Hough Space. From this, if we were to map all the edge points from an edge image onto the Hough Space, it will generate a lot of cosine curves. If two edge points lay on the same line, their corresponding cosine curves will intersect each other on a specific (ρ, θ) pair. Thus, the Hough Transform algorithm detects lines by finding the (ρ, θ) pairs that have a number of intersections larger than a certain threshold. It is worth noting that this method of thresholding might not always yield the best result without doing some preprocessing like neighborhood suppression on the Hough Space to remove similar lines in the edge image.

5.6 Object Identification

Object identification involves recognizing and labeling objects within an image or video.

Methods of Object Identification:

- Template Matching – Compares an input image with stored templates.
- Feature-Based Recognition – Uses key features to identify objects.
- Deep Learning (CNNs, YOLO) – Advanced AI techniques for real-time detection.

Unit-6

Expert system

6.1Introduction to expert system

Expert System

Expert systems are advanced computer programs designed to mimic human expertise in specific domains. They analyze data, solve complex problems, and provide insights, significantly impacting various industries, including healthcare, finance, and engineering.

Introduction to Expert Systems

Expert systems are computer programs emulating human expert decision-making. Developed in the 1960s, they combine a knowledge base, inference engine, and user interface. Examples include MYCIN and Deep Blue. They have revolutionized medicine, engineering, and finance.

Performance

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

Characteristics of Expert Systems

- High Performance: Solves complex problems efficiently and accurately.

- Understandable: Accepts and provides output in human language.

- Reliable: Produces accurate results.

- Highly responsive: Delivers fast results.

Components of Expert System

- User Interface: Enables interaction with users.

- Inference Engine: Derives conclusions using rules.

- Knowledge Base: Stores domain-specific knowledge.

Inference Engine

Acts as the brain of the expert system. Types:

- Deterministic: Based on facts and rules.

- Probabilistic: Contains uncertainty.

Modes:

- Forward Chaining: Starts from known facts.

- Backward Chaining: Starts from goal and works backward.

Knowledge Base

Stores domain-specific knowledge.

- Factual Knowledge: Based on facts.

- Heuristic Knowledge: Based on experience.

- Representation: If-else rules.

- Acquisition: Extracting and structuring domain knowledge.

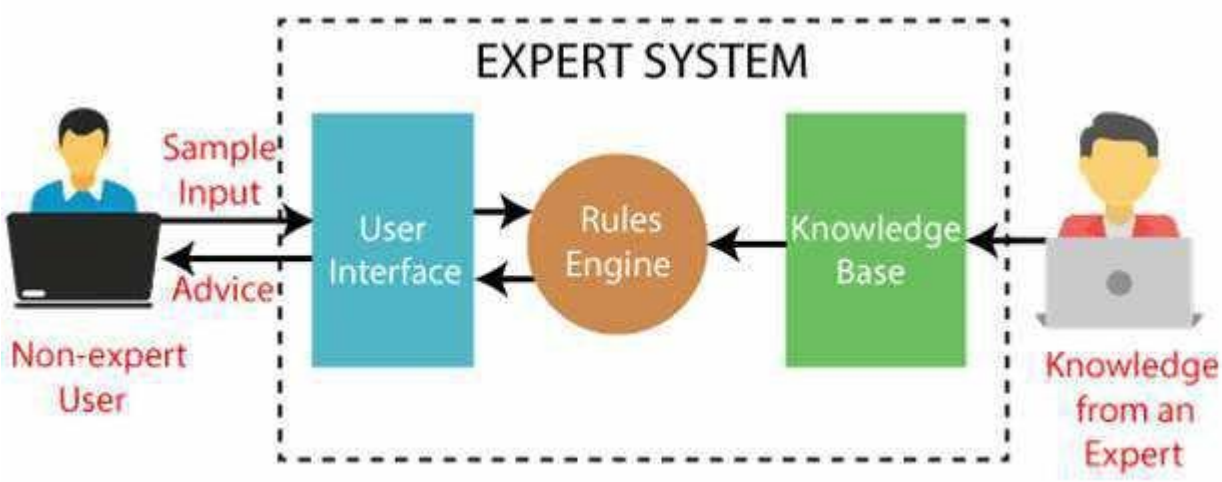
What is an Expert System?

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

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

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

Below is the block diagram that represents the working of an expert system:



Note: It is important to remember that an expert system is not used to replace the human experts; instead, it is used to assist the human in making a complex decision. These systems do not have human capabilities of thinking and work on the basis of the knowledge base of the particular domain.

Below are some popular examples of the Expert System:

- **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.
- **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.
- **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer at early stages.
-

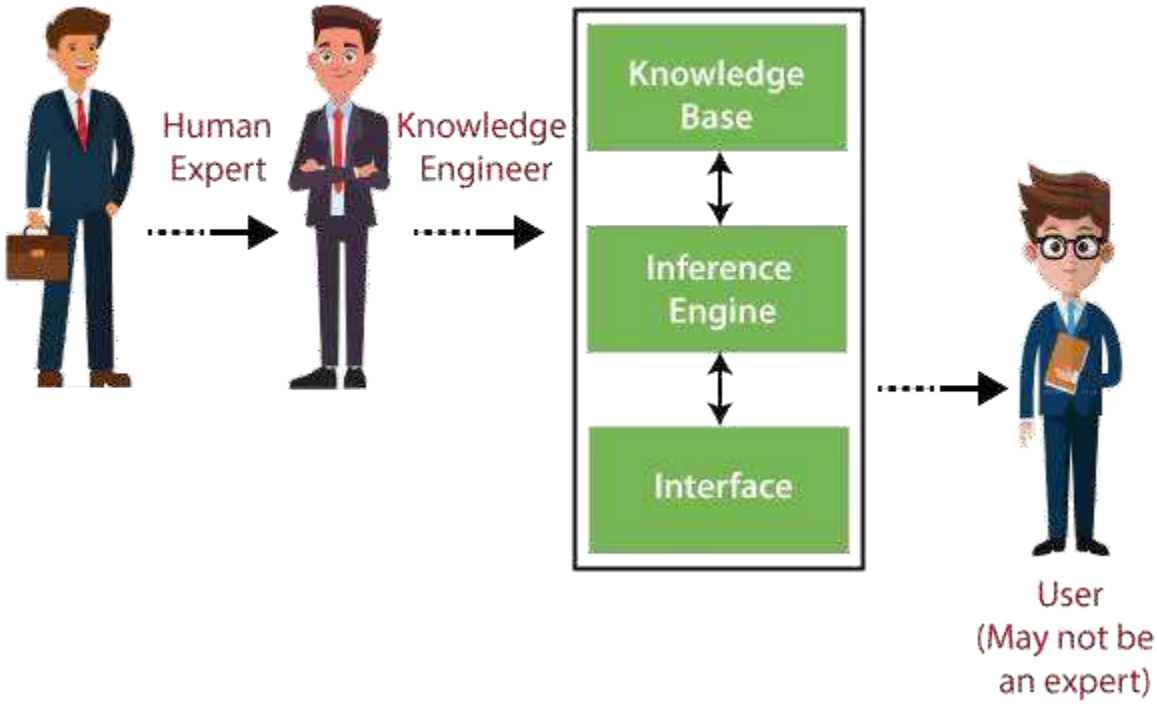
Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Components of Expert System

An expert system mainly consists of three components:

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



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. 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(Rules of 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. It helps in deriving an error-free solution of queries asked by the user. ○ With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- There are two types of inference engine:
- **Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.
- **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **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 acquired 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 and rules of a particular domain or subject. ○ One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

- **Factual Knowledge:** The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.
- **Heuristic Knowledge:** This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: 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 the knowledge base.

6.2Basic Architecture of expert system

Typical expert system architecture is shown in Figure 11.1.

The knowledge base contains the specific domain knowledge that is used by an expert to derive conclusions from facts.

In the case of a rule-based expert system, this domain knowledge is expressed in the form of a series of rules.

The explanation system provides information to the user about how the inference engine arrived at its conclusions. This can often be essential, particularly if the advice being given is of a critical nature, such as with a medical diagnosis system.

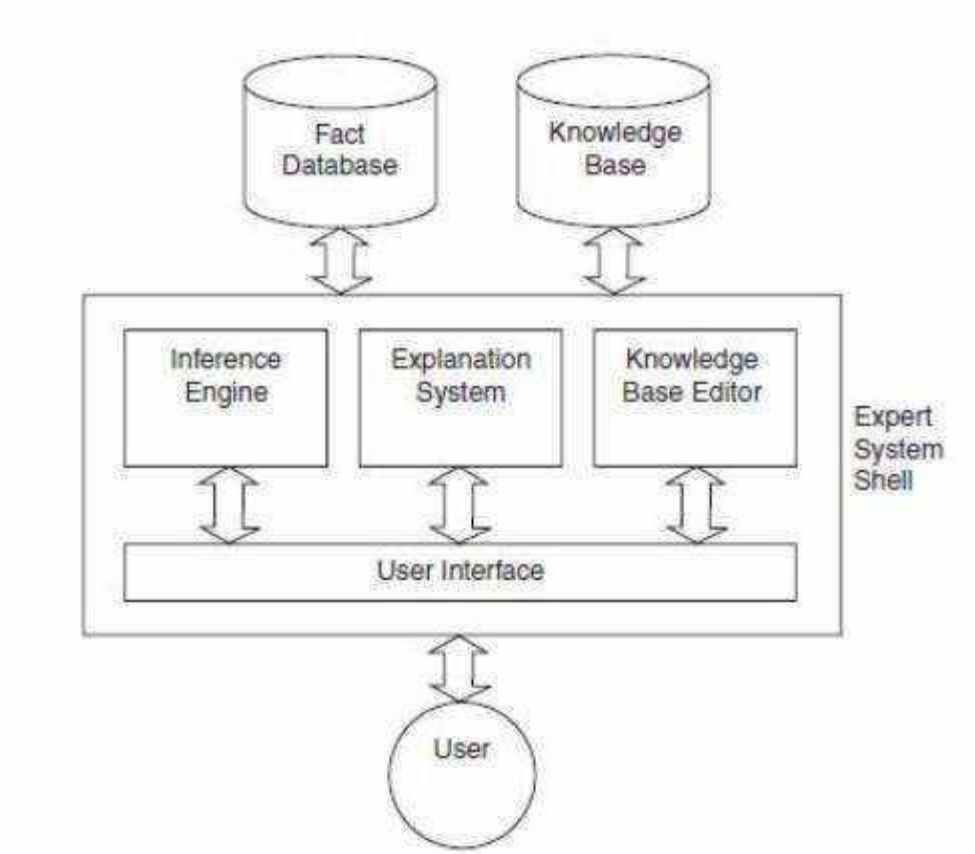


Fig Expert System Architecture

If the system has used faulty reasoning to arrive at its conclusions, then the user may be able to see this by examining the data given by the explanation system.

The fact database contains the case-specific data that are to be used in a particular case to derive a conclusion.

In the case of a medical expert system, this would contain information that had been obtained about the patient's condition.

The user of the expert system interfaces with it through a user interface, which provides access to the inference engine, the explanation system, and the knowledge-base editor.

The inference engine is the part of the system that uses the rules and facts to derive conclusions. The inference engine will use forward chaining,

backward chaining, or a combination of the two to make inferences from the data that are available to it.

The knowledge-base editor allows the user to edit the information that is contained in the knowledge base.

> The knowledge-base editor is not usually made available to the end user of the system but is used by the knowledge engineer or the expert to provide and update the knowledge that is contained within the system.

6.3Types of problem solved by expert system

The expert systems are capable of −

- Advising
- Instructing and assisting a human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion

6.4Features of an expert system

- Human experts are perishable, but an expert system is permanent.
- It helps to distribute the expertise of a human.
- One expert system may contain knowledge from more than one human experts thus making the solutions more efficient.

- It decreases the cost of consulting an expert for various domains such as medical diagnosis.
- They use a knowledge base and inference engine.
- Expert systems can solve complex problems by deducing new facts through existing facts of knowledge, represented mostly as if-then rules rather than through conventional procedural code.
- Expert systems were among the first truly successful forms of artificial intelligence (AI) software.

6.5 Expert system architecture

Expert architecture is internally structure that represents to the knowledge base has the certain domain knowledge that is implemented by an expert to server conclusion from facts.

The architecture of an expert system can be better understood by comparing it with a conventional programming.

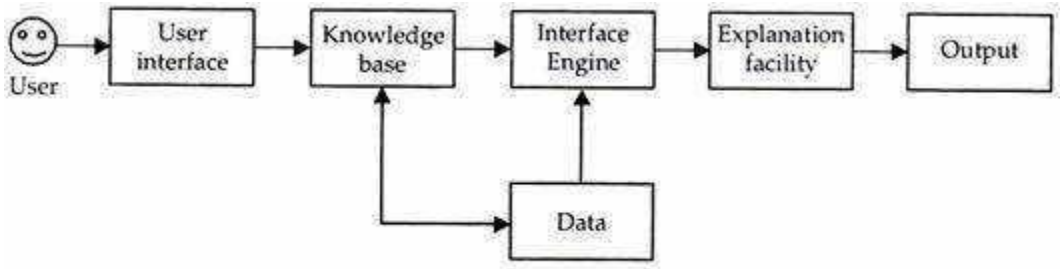


Fig. 12.3(a) *Expert System.*

The main components in an Expert System Software are knowledge base, inference engine including explanation facility, user interface mechanism and interfacing, whereas the main components of a conventional software are data (database), program code, interpreter/compiler though not obvious to the user and sparse user-interface.

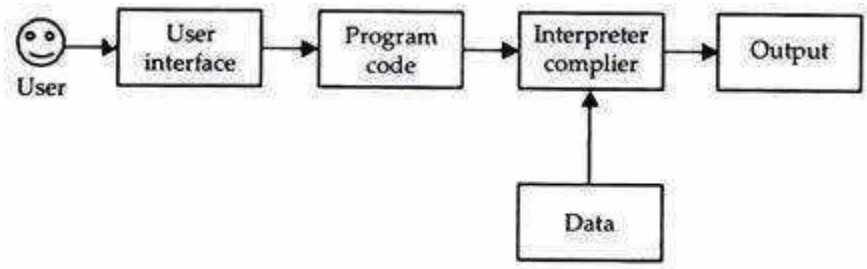


Fig. 12.3(b) *Conventional Software program.*

6.6 Expert system tools

The expert system tools are programming systems which simplify the job of constructing an expert system. They range from very high-level programming language to low-level support facilities. We divide expert system tools into four major categories as shown in Fig. 12.12. We will describe these tools and explain briefly how they are used.

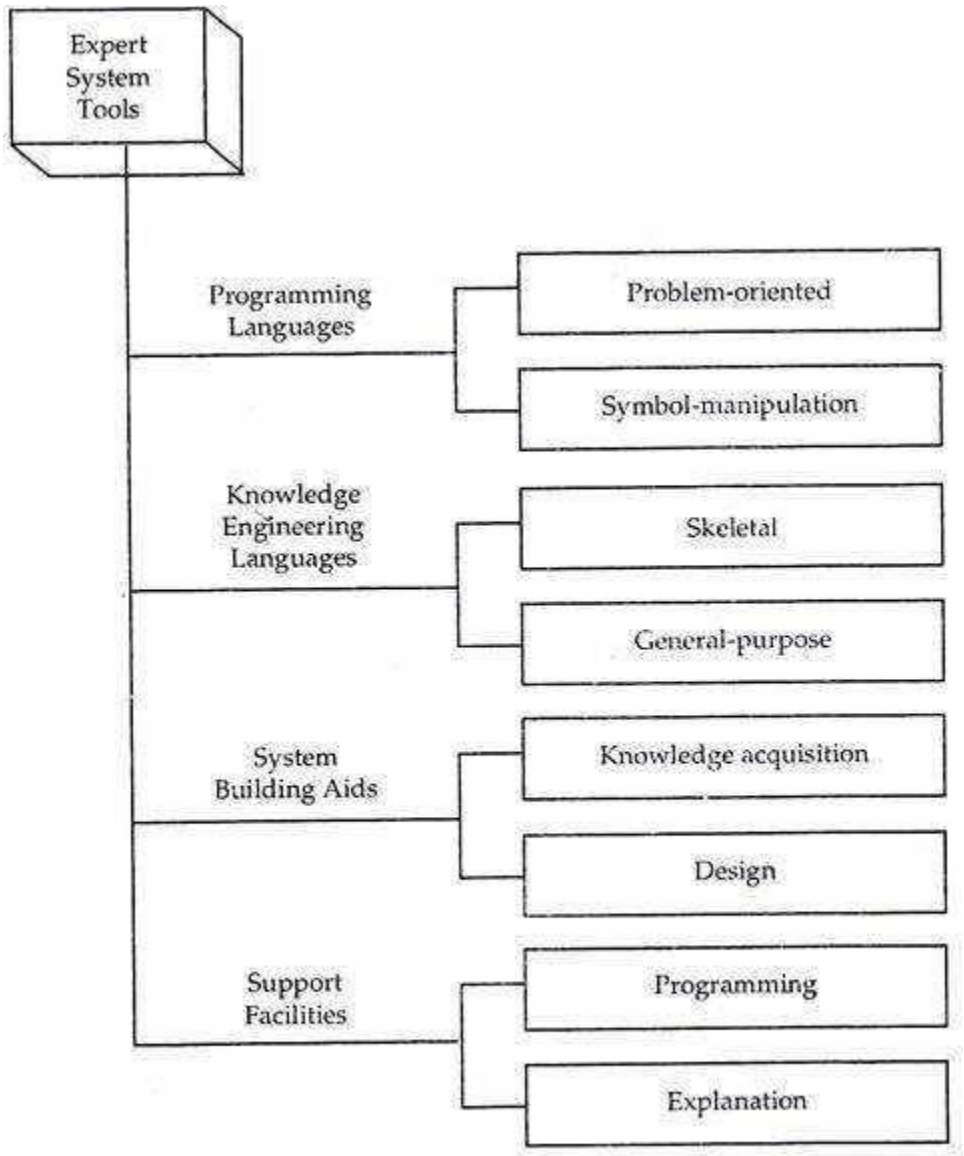


Fig. 12.12. *Types of tools available for expert system building*

6.7 Existing expert systems

There are mainly five types of expert systems. They are **rule based expert system, frame based expert system, fuzzy expert system, neural expert system and neuro-fuzzy expert system.**

6.8 Applications of expert system technology

Some popular Application of Expert System:

- Information management
- Hospitals and medical facilities
- Help desks management
- Employee performance evaluation
- Loan analysis
- Virus detection
- Useful for repair and maintenance projects
- Warehouse optimization Planning and scheduling
- The configuration of manufactured objects
- Financial decision making Knowledge publishing
- Process monitoring and control
- Supervise the operation of the plant and controller
- Stock market trading
- Airline scheduling & cargo schedules

Development Trend

- Core Objective: Mimic expert reasoning.

- Evolved since the 1960s with AI development.

Applications

- Medicine: Diagnosis, treatment planning.

- Finance: Credit scoring, fraud detection.

- Engineering: Design optimization.

- Manufacturing: Quality control, scheduling.

Tools

- Programming Languages: LISP, Prolog, Python.

- Development Shells: CLIPS, JESS.

Case Studies

- MYCIN: Diagnosed infections.

- DENDRAL: Elucidated chemical structures.

- Deep Blue: Chess champion.

- XCON/R1: Configured computer systems.

Advantages and Disadvantages

- Consistency, Availability, Cost-Effective.

- Limitations in knowledge acquisition, common sense.