# ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

**By**

**Dr. S. Rethinavalli,**
**Ms.V.Manibabu,**
**Ms. Viji Parthasarathy**


**Ryan Publishers**

**ISBN : 978-93-341-5454-2**

**Publisher : Ryan Publishers**
**15, Sivan Koil 1st Street,**
**Banker's colony Extension,**
**Kumaran Nagar,**
**Trichy - 620017,**
**Tamil Nadu, India.**
**Website: www.ryanpublishers.co.in**
**Email: editor@ryanpublishers.co.in**
**Ph: +91 6374561101**

**Price: Rs. 120\-**

## ABOUT THE AUTHOR

**Dr.S.Rethinavalli**
M.C.A., M.Phil.,Ph.D.,
Academic Director

**Dr. S. Rethinavalli**, with over 24 years of teaching experience and 10 years in research, serves as the Academic Director of the MCA Department at Shrimati Indira Gandhi College. She has guided mor than 10 research students, published 16 papers in esteemed journals, and obtained six patents. Her excellence in research earned her the Best Researcher Award in 2024. As a member of the Board of Studies for PG Computer Science at Bharathidasan University. she is committed to academic leadership and student development. Dr. Rethinavalli continues to inspire and guide the next generation of computer science professionals with her passion for research, innovation, and excellence in education.

Ms.V.Manibabu
M.Sc(Maths).,M.Phil.,
PGDCA.,MCA.,M.Phi
l(CS).,Ph.D(doing)
Assistant Professor

**Ms.V.Manibabu** is a seasoned educator and researcher with 23 years of distinguished service in the field of Computer Sciences. Currently, serving as a faculty member at Shrimati Indira Gandhi college Trichy, a renowned private institution, Ms.V.Manibabu has consistently demonstrated exceptional teaching and mentoring skills, guiding numerous students to achieve academic excellence. A distinguished academic, Ms.V.Manibabu is pursuing Ph.D. in Computer Science, currently in the second year. With an impressive research portfolio, Ms.V.Manibabu has presented and published numerous papers in esteemed conferences, showcasing expertise in cutting-edge technologies.Throughout their illustrious career, Ms.V.Manibabu has effectively handled various

papers, producing outstanding results and earning recognition for innovative teaching methodologies. This book, tailored to student syllabi, embodies Ms.V.Manibabu's rich experience and passion for empowering the next generation of computer science professionals.

**Ms.Vijiparthasarsathy**
M.Sc(IT).,M.Phil.,MCA
Assistant Professor

**Ms. Viji Parthasarathy** is a dedicated Assistant Professor with extensive experience in the Department of Computer Applications at Shrimati Indira Gandhi College. She has actively participated in professional development through various orientation and faculty programs, presented more than five papers at international conferences, and published research articles on current advancements in her field. Currently, she is pursuing a Ph.D. in Computer Science. Recognized for her expertise, Ms. Viji Parthasarathy has received several honors, including the Best Teacher and Best Women Faculty awards, and she was a top 100 finalist in the Global Best Teachers Awards for the 2024–2025 cycle. She holds six AI-based patents from India and the UK, has completed certificate courses in advanced technologies, and has authored books on Java, Python, and Artificial Intelligence & Machine Learning.

## Introduction

Artificial Intelligence (AI) has emerged as one of the most transformative forces of our time, reshaping industries, enhancing productivity, and redefining the ways we interact with technology. From virtual assistants and self-driving cars to sophisticated recommendation systems, AI is integrated into many aspects of our daily lives. Its potential to analyze vast amounts of data, learn from experience, and make decisions autonomously positions it as a cornerstone of modern innovation.

Expert Systems, a subset of AI, have played a crucial role in this evolution. These systems emulate the decision-making abilities of a human expert, utilizing knowledge bases and inference engines to solve complex problems across various fields. They have been instrumental in sectors like healthcare, finance, and engineering, where they assist professionals in making informed decisions based on data-driven insights.

This book aims to provide a thorough exploration of both AI and Expert Systems, grounded in the curriculum of BDU. It covers fundamental concepts, algorithms, and methodologies, while also addressing practical applications and challenges faced in the field.

Throughout this text, you will find:

**Foundational Concepts**: An introduction to AI, its history, and key components, laying the groundwork for deeper exploration.

**Core Technologies**: Detailed discussions on machine learning, natural language processing, computer vision, and more, explaining how these technologies drive AI application

**Expert Systems**: Insights into the architecture and functionality of expert systems, including knowledge representation, reasoning, and learning mechanism

**Real-World Applications**: Case studies and examples that illustrate how AI and expert systems are being used to solve real-world problems, demonstrating their impact across various sectors.

**Ethical Considerations**: A critical examination of the ethical implications and societal impacts of AI, encouraging readers to think critically about the technology they are engaging with.

This book is designed for students, educators, and professionals seeking a comprehensive understanding of AI and Expert Systems. Whether you are new to the field or looking to deepen your knowledge, we hope this text serves as a valuable resource on your journey through the fascinating world of artificial intelligence.

## PREFACE

In the rapidly evolving landscape of technology, the significance of Artificial Intelligence (AI) and Expert Systems cannot be overstated. This book is crafted with the intent to provide a comprehensive and accessible resource for students, educators, and professionals who are keen to explore the principles, applications, and future of these transformative fields.

Rooted in the syllabus of BDU, this book is designed to bridge theoretical concepts with practical implementations. It addresses foundational topics such as machine learning, natural language processing, and the architecture of expert systems, while also delving into advanced applications across various domains, including healthcare, finance, and robotics.

We recognize that AI is not merely a collection of algorithms and data; it is a field that blends creativity and analytical thinking. Therefore, this book emphasizes not only the technical skills necessary for developing AI solutions but also the ethical considerations and societal impacts that come with them.

Our aim is to foster a deep understanding of AI and expert systems, equipping readers with the knowledge and tools to innovate and contribute meaningfully to this exciting field. Each chapter is structured to build upon the last, encouraging a step-by-step progression through complex topics, accompanied by real-world examples and hands-on exercises.

We hope this book serves as a valuable guide on your journey through the world of Artificial Intelligence and Expert Systems, inspiring you to push boundaries and explore the limitless possibilities of technology.

Happy learning

Authors
**Dr.S.Rethinavalli,**
**Ms.V.Manibabu**
**Ms.Viji Parthasarathy**
Department of Computer Applications
Shrimati Indira Gandhi College ,Trichy

## AI SYLLABUS

**UNIT-I**

Problems and search-searching strategies-uninformed search-breadth first search,depth first search,uniform cost search,depth limited search,iterractive deepening search,bidirectional search-informed search-best first search-greedy best-first search,A*search-constraint satisfaction problem,local searching strategies.

**UNIT-II**

Reasoning- Symbolic Reasoning under uncertainity- statistical Reasoning -week slot and filler structure - semantic nets- Frames, strong slot and Filler structure -conceptual dependency –scripts- cyc

**UNIT-III**

Knowledge Representation- knowledge representation issue- using predicate logic-representing knowledge using rules, syntatic- semantic of Representation, logic and slot filler- Game playing Minimal search,- Alpha Beta cuttoff- iterative Deepening planning- component of planning system - Goal stack planning.

**UNIT IV**

NP (Natural language processing)- Syntatic processes, semantic analysis -parallel and distributed AI psychological modelling ,parallelism and distributed in reasoning system ,learning connectionist model-hopefield networks, neural networks.

**UNIT V**

Expert system, common sense -Qualitative physics, common sense ontologist, Memory organization, expert system, expert system cell explanation.

## CONTENT

# UNIT 1

## Problem:

A problem is a particular task or challenge that calls for decision-making orsolution-finding. In artificial intelligence, an issue is simply a task that needs to be completed; these tasks can be anything from straightforward math problems to intricate decision-making situations. Artificial intelligence encompasses various jobs and challenges, from basic math operations to sophisticated ones like picture recognition, natural language processing, game play, and optimization. Every problem has a goal state that must be attained, a defined set of initial states, and potential actions or moves.

## Search Algorithms in Artificial Intelligence:

## Introduction:

Search algorithms in AI are the algorithms that are created to aid the searchers in getting the right solution. The search issue contains search space, first start and end point. Now by performing simulation of scenarios and alternatives, searching algorithms help AI agents find the optimal state for the task.

Logic used in algorithms processes the initial state and tries to get the expected state as the solution. Because of this, AI machines and applications just functioning using search engines and solutions that come from these algorithms can only be as effective as the algorithms.

AI agents can make the AI interfaces usable without any software literacy. The agents that carry out such activities do so with the aim of reaching an end goal and develop action plans that in the end will bring the mission to an end. Completion of the action is gained after the steps of these different actions. The AI-agents finds the best way through the process by evaluating all the alternatives which are present. Search systems are a common task in artificial intelligence by which you are going to find the optimum solution for the AI agents.

**Problem-solving agents:**

In Artificial Intelligence, Search techniques are universal problem-solving methods. Rational agents or Problem-solving agents in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result. Problem-solving agents are the goal-based agents and use atomic representation. In this topic, we will learn various problem-solving search algorithms.

**Search Algorithm Terminologies:**

- **Search:** Searching is a step by step procedure to solve a search-problemin a given search space. A search problem can have three main factors:
    1. **Search Space:** Search space represents a set of possible solutions,which a system may have.
    2. **Start State:** It is a state from where agent begins **the search**.
    3. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding tothe initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Path Cost:** It is a function which assigns a numeric cost to

each path.

- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.Properties of Search Algorithms:
- Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

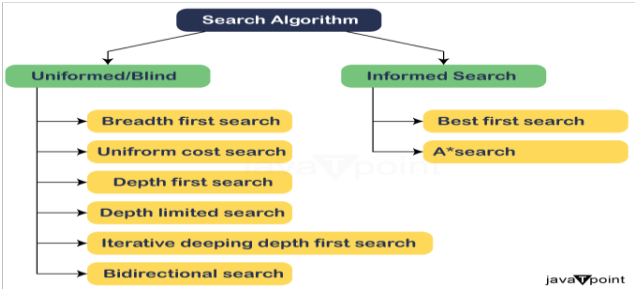
**Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

**Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

**Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.

**Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

**Types of search algorithms:**



**Uninformed/Blind Search:**

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search. It examines each node of the tree until it achieves the goal node.

**It can be divided into six main types:**

- Breadth-first search
- Uniform cost search
- Depth-first search
- Depth limited search
- Iterative deepening depth-first search

- **Bidirectional Search Informed Search**

Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.

A heuristic is a way which might not always be guaranteed for best solutions butguaranteed to find a good solution in reasonable time.

Informed search can solve much complex problem which could not be solved inanother way.

An example of informed search algorithms is a traveling salesman problem:

- Greedy Search

- A* Search

## 1. Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

**Advantages:**

- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.
- It also helps in finding the shortest path in goal state, since it needs all nodes at the same hierarchical level before making a move to nodes at lower levels.
- It is also very easy to comprehend with the help of this we can assign thehigher rank among path types.

**Disadvantages:**

**Advertisement**

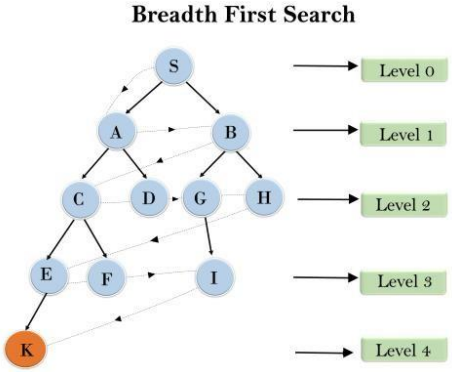
- It requires lots of memory since each level of the tree must be saved intomemory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

- It can be very inefficient approach for searching through deeply layered spaces, as it needs to thoroughly explore all nodes at each level before moving on to the next.

**Example:**

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

1. S---> A--->B---->C--->D---->G--->H--->E---->F---->I >K




**Time Complexity:** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$T(b) = 1+b^2+b^3+ \quad + b^d = O(b^d)$

**Space Complexity:** Space complexity of BFS algorithm is given by theMemory size of frontier which is $O(b^d)$.

**Completeness:** BFS is complete, which means if the shallowest goal node is atsome finite depth, then BFS will find a solution.

**Optimality:** BFS is optimal if path cost is a non-decreasing function of thedepth of the node.

## 2. Depth-first Search

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

**Note:** Backtracking is an algorithm technique for finding all possible solutionsusing recursion.

**Advantage:**

- DFS requires very less memory as it only needs to store a stack of thenodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if ittraverses in the right path).
- With the help of this we can stores the route which is being tracked inmemory to save time as it only needs to keep one at a particular time.

**Disadvantage:**

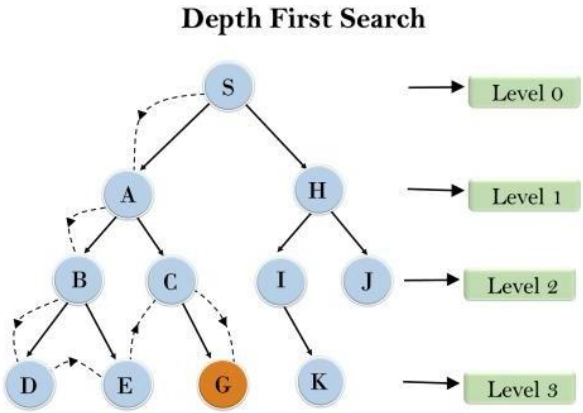
- There is the possibility that many states keep re-occurring, and there is noguarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go tothe infinite loop.
- The depth-first search (DFS) algorithm does not always find the shortest path to a solution.

**Example:**

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

Root node--->Left node----------> right node.

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




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

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

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

**Where, m= maximum depth of any node and this can be much larger thand (Shallowest solution depth)**

**Space Complexity:** DFS algorithm needs to store only single path

from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

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

## 3. Depth-Limited Search Algorithm:

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinitepath in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

**Depth-limited search can be terminated with two Conditions of failure:**

- **Standard failure value:** It indicates that problem does not have anysolution.
- **Cutoff failure value:** It defines no solution for the problem within agiven depth limit.

**Advantages:**

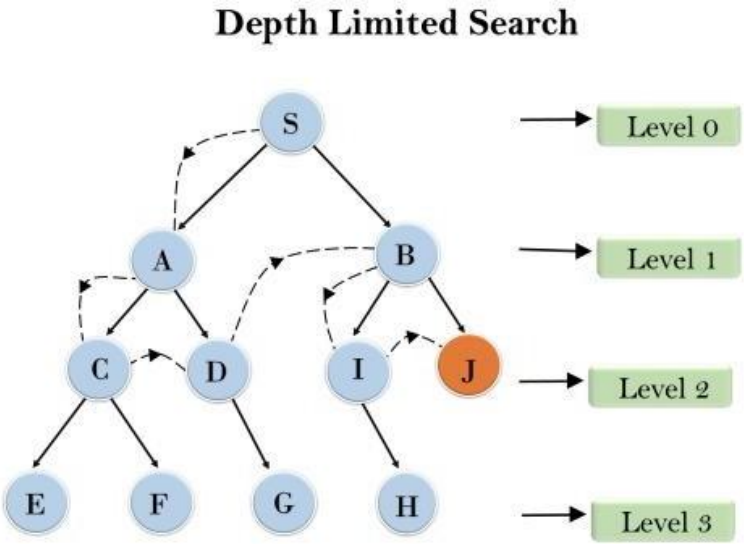
- Depth-Limited Search will restrict the search depth of the tree, thus, the algorithm will require fewer memory resources than the straight BFS (Breadth-First Search) and IDDFS (Iterative Deepening Depth-First Search). After all, this implies automatic selection of more segments of the search space and the consequent why consumption of the resources. Due to the depth restriction, DLS omits a predicament of holding the entire search tree within memory which contemplatively leaves room for a more memory-efficient vice for solving a particular kind of problems.
- When there is a leaf node depth which is as large as the highest level allowed, do not describe its children, and then discard it from the stack.
- Depth-Limited Search does not explain the infinite loops

which can arisein classical when there are cycles in graph of cities.

**Disadvantages:**

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.
- The effectiveness of the Depth-Limited Search (DLS) algorithm is largelydependent on the depth limit specified. If the depth limit is set too low, the algorithm may fail to find the solution altogether.

**Example:**




**Completeness:** DLS search algorithm is complete if the solution is above thedepth-limit.

**Time Complexity:** Time complexity of DLS algorithm is **$O(b^{\ell})$** where b is the branching factor of the search tree, and l is the depth limit.

**Space Complexity:** Space complexity of DLS algorithm is **$O(b \times \ell)$** where **b** is the branching factor of the search tree, and **l** is the depth limit.

**Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if $\ell > d$.

**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:**

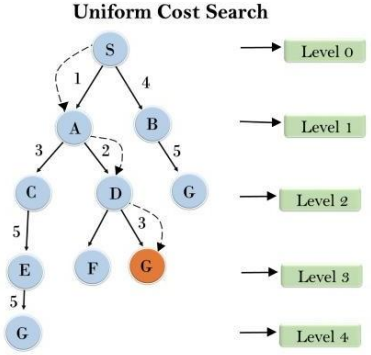
- Uniform cost search is optimal because at every state the path with the least cost is chosen.
- It is an efficient when the edge weights are small, as it explores the paths in an order that ensures that the shortest path is found early.
- It's a fundamental search method that is not overly complex, making it accessible for many users.
- It is a type of comprehensive algorithm that will find a solution if one exists. This means the algorithm is complete, ensuring it can locate a solution whenever a viable one is available. The algorithm covers all the necessary steps to

arrive at a resolution.

**Disadvantages:**

- 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 inan infinite loop.
- When in operation, UCS shall know all the edge weights to start off the search.
- This search holds constant the list of the nodes that it has already discovered in a priority queue. Such is a much weightier thing if you havea large graph. Algorithm allocates the memory by storing the path sequence of prioritizes, which can be memory intensive as the graph gets larger.With the help of Uniform cost search we can end up with the problem if the graph has edge's cycles with smaller cost than that of the shortest path.
- The Uniform cost search will keep deploying priority queue so that the paths explored can be stored in any case as the graph size can be even bigger that can eventually result in too much memory being used.

**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.

## 4. Iterative deepeningdepth-first Search:

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

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

This Search algorithm combines the benefits of Breadth-first search's fast searchand depth-first search's memory efficiency.

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

**Here are the steps for Iterative deepening depth first search algorithm:**

- Set the depth limit to 0.
- Perform DFS to the depth limit.
- If the goal state is found, return it.
- If the goal state is not found and the maximum depth has not been reached, increment the depth limit and repeat steps 2-4.
- If the goal state is not found and the maximum depth has been reached,terminate the search and return failure.

**Advantages:**

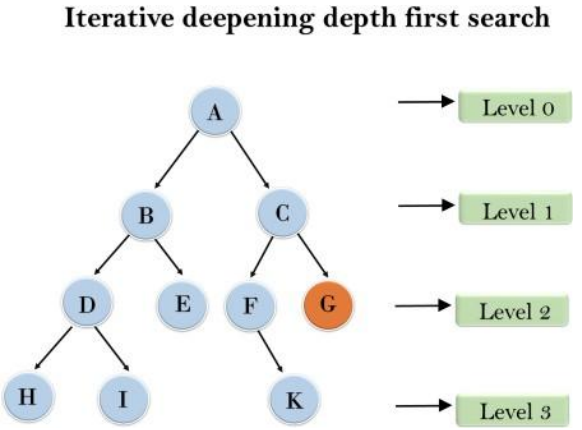
- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.
- It is a type of straightforward which is used to put into practice since it builds upon the conventional depth-first search algorithm.
- It is a type ofsearch algorithm which provides guarantees to find the optimal solution, as long as the cost of each edge in the search space isthe same.
- It is a type ofcomplete algorithm, and the meaning of this is it will alwaysfind a solution if one exists.
- The Iterative Deepening Depth-First Search (IDDFS) algorithm uses less memory compared to Breadth-First Search (BFS) because it only stores the current path in memory, rather than the entire search tree.

**Disadvantages:**

- The main drawback of IDDFS is that it repeats all the work of theprevious phase.

**Example**:

Following tree structure is showing the iterative deepening depth-first search. IDDFS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:




| | | | | | | |
|---|---|---|---|---|---|---|
| 1st | | Iteration--> | | | | A |
| 2nd | Iteration-----> | A, | | | B, | C |
| 3rd | Iteration-->A, | B, | | E, | C, | G |
| 4th | Iteration-->A | D, | I, | E, | C, | G |

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

**Completeness:**

This algorithm is complete is if the branching factor is finite.

**Time Complexity:**

Let's suppose b is the branching factor and depth is d then the worst-case timecomplexity is $\mathbf{O(b^d)}$.

**Space Complexity:**

The space complexity of IDDFS will be **O(bd)**.**Optimal:**

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

## 5. Bidirectional Search Algorithm:

Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.

**Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.**

**Advantages:**

- Bidirectional search is fast.
- Bidirectional search requires less memory
- The graph can be extremely helpful when it is very large in size and there is no way to make it smaller. In such cases, using this tool becomes particularly useful.
- The cost of expanding nodes can be high in certain cases. In such scenarios, using this approach can help reduce the number of nodes that need to be expanded.

**Disadvantages:**

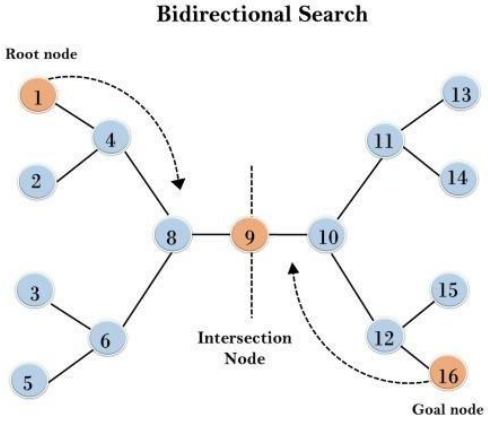
- Implementation of the bidirectional search tree is difficult.
- In bidirectional search, one should know the goal state in advance.
- Finding an efficient way to check if a match exists between

search trees can be tricky, which can increase the time it takes to complete the task.

**Example:**

In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.

The algorithm terminates at node 9 where two searches meet.




**Completeness:** Bidirectional Search is complete if we use BFS in bothsearches.

**Time Complexity:** Time complexity of bidirectional search using BFS is **O($b^d$)**. **Space Complexity:** Space complexity of bidirectional search is **O($b^d$)**.

**Optimal:** Bidirectional search is Optimal.

**Greedy Algorithm**

A **greedy algorithm** is a problem-solving technique that makes the

best local choice at each step in the hope of finding the global optimum solution. It prioritizes immediate benefits over long-term consequences, making decisions based on the current situation without considering future implications. While this approach can be efficient and straightforward, it doesn't guarantee the best overall outcome for all problems.

However, it's important to note that not all problems are suitable for greedyalgorithms. They work best when the problem exhibits the following properties:

- **Greedy Choice Property:** The optimal solution can be constructed bymaking the best local choice at each step.
- **Optimal Substructure:** The optimal solution to the problem contains theoptimal solutions to its subproblems.

**Characteristics of Greedy Algorithm**

Here are the characteristics of a greedy algorithm:

- Greedy algorithms are simple and easy to implement.
- They are efficient in terms of time complexity, often providing quicksolutions.
- Greedy algorithms are used for optimization problems where a **locally optimal** choice leads to a **globally optimal** solution.
- These algorithms do not reconsider previous choices, as they make decisions based on current information without looking ahead.
- Greedy algorithms are suitable for problems for optimal substructure.

**A*Search:**

**A* Search** is an informed best-first search algorithm that efficiently determines the lowest cost path between any two nodes in a directed weighted graph with non-negative edge weights. This algorithm is a

variant of Dijkstra's algorithm. A slight difference arises from the fact that an evaluation function is used to determine which node to explore next.

**Evaluation Function**

The evaluation function, **f(x)**, for the A* search algorithm is the following:$f(x) = g(x) + h(x)$

Where **g(x)** represents the cost to get to node **x** and **h(x)** represents the estimatedcost to arrive at the goal node from node **x**.

For the algorithm to generate the correct result, the evaluation function must be **admissible**, meaning that it never overestimates the cost to arrive at the goal node.

**The A* Algorithm:**

The A* algorithm is implemented in a similar way to Dijkstra's algorithm. Given a weighted graph with non-negative edge weights, to find the lowest-cost path from a start node **S** to a goal node **G**, two lists are used:

- An **open list**, implemented as a priority queue, which stores the next nodes to be explored. Because this is a priority queue, the most promising candidate node (the one with the lowest value from the evaluation function) is always at the top. Initially, the only node in this list is the start node **S**.
- A **closed list** which stores the nodes that have already been evaluated. When a node is in the closed list, it means that the lowest-cost path to thatnode has been found.

To find the lowest cost path, a search tree is constructed in the following way:

1. Initialize a tree with the root node being the start node **S**.
2. Remove the top node from the open list for exploration.
3. Add the current node to the closed list.

4. Add all nodes that have an incoming edge from the current node as child nodes in the tree.
5. Update the lowest cost to reach the child node

current_lowest_cost = min(current_lowest_cost, parent_node_cost + edge_weight)

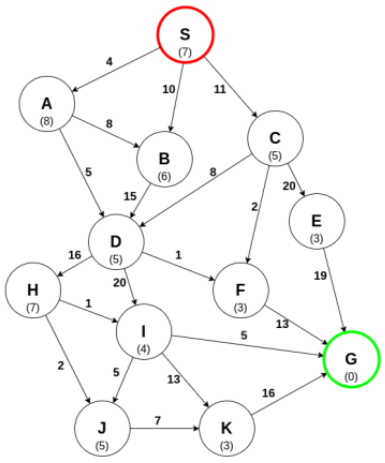
All nodes except for the start node start with a lowest cost of infinity. The start node has an initial lowest cost of 0.

The algorithm concludes when the goal node **G** is removed from the open list and explored, indicating that a shortest path has been found. The shortest path isthe path from the start node **S** to the goal node **G** in the tree.

Note: The algorithm ends when the goal node **G** has been explored, NOT when it is added to the open list.

**Example**

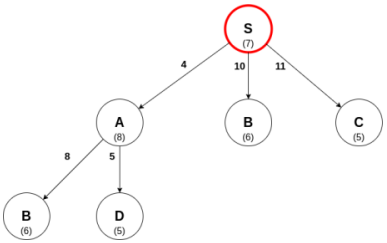
Consider the following example of trying to find the shortest path from **S** to **G**



Each edge has an associated weight, and each node has a heuristic cost (inparentheses).

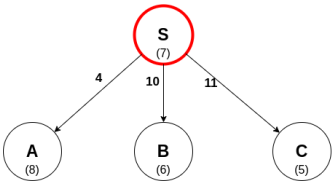
An open list is maintained in which the node **S** is the only node in the list. Thesearch tree can now be constructed.

Exploring **S**:




| Node[cost] |
|---|
| D[14] |
| C[16] |
| B[16] |
| B[18] |

| Closed List |
|---|
| S |
| A |

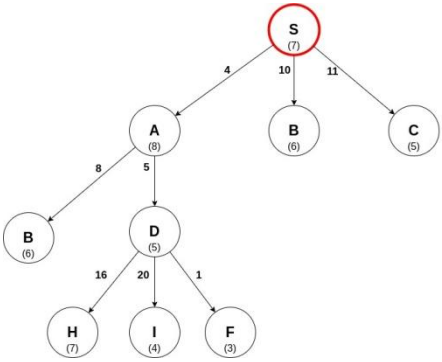



| Node[cost] |
|---|
| A[12] |
| B[16] |
| C[16] |

| Closed List |
|---|
| S |

**A** is the current most promising path, so it is explored next:

Exploring **D**:



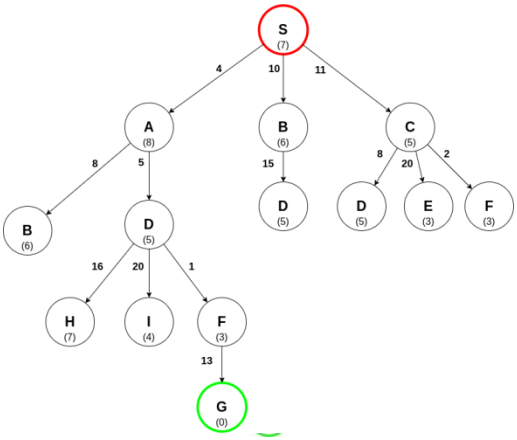

| Node[cost] |
|---|
| F[13] |
| C[16] |
| B[16] |
| H[32] |
| I[33] |
| B[18] |

| Closed List |
|---|
| S |
| A |
| D |

Exploring **F**:

Notice that the goal node **G** has been found. However, it hasn't been explored, so the algorithm continues because there may be a shorter path to G. Thenode **B** has two entries in the open list: one at a cost of 16 (child of **S**) and oneat a cost of 18 (child of **A**). The one with the lowest cost is explored next:

Exploring **C**:

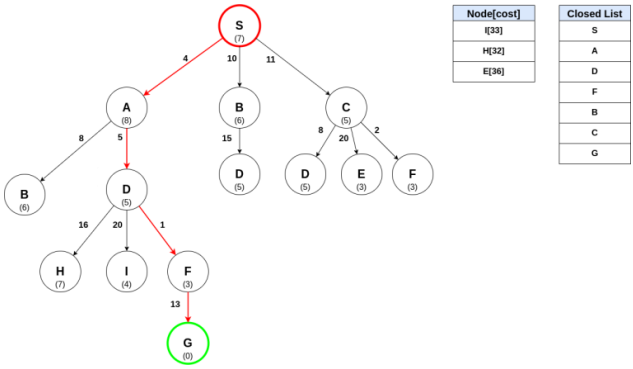



| Node[cost] |
|---|
| G[23] |
| I[33] |
| H[32] |
| E[36] |

| Closed List |
|---|
| S |
| A |
| D |
| F |
| B |
| C |

The next node in the open list is again **B**. However, because **B** has already been explored, meaning the shortest path to **B** has been found, it is not explored again, and the algorithm continues to the next candidate.

The next node to be explored is the goal node **G**, meaning the shortest path to **G** has been found! The path is constructed by tracing the graph backwards from **G** to **S**:




**Constraint Satisfaction Problem (CSP):**

A **Constraint Satisfaction Problem** is a mathematical problem where the solution must meet a number of constraints. In a CSP, the objective is to assign values to variables such that all the constraints are satisfied. CSPs are used extensively in artificial intelligence for decision-making problems where resources must be managed or arranged within strict guidelines.

**Common applications of CSPs include:**

- **Scheduling:** Assigning resources like employees or equipment while respecting time and availability constraints.
- **Planning:** Organizing tasks with specific deadlines or sequences.
- **Resource Allocation:** Distributing resources efficiently without overuse.

**Components of Constraint Satisfaction Problems**

**CSPs are composed of three key elements:**

1. **Variables:** The things that need to be determined are variables. Variablesin a CSP are the objects that must have values assigned to them in order to satisfy a particular set of constraints. Boolean, integer, and categorical variables are just a few examples of the various types of variables, for instance, could stand in for the many puzzle cells that need to be filled with numbers in a sudoku puzzle.

2. **Domains:** The range of potential values that a variable can have is represented by domains. Depending on the issue, a domain may be finite or limitless. For instance, in Sudoku, the set of numbers from 1 to 9 can serve as the domain of a variable representing a problem cell.

3. **Constraints:** The guidelines that control how variables relate to one another are known as constraints. Constraints in a CSP define the ranges of possible values for variables. Unary constraints, binary constraints, and higher-order constraints are only a few examples of the various sorts of constraints. For instance, in a sudoku problem, the restrictions might be that each row, column, and 3×3 box can only have one instance of each number from 1 to 9.

**Types of Constraint Satisfaction Problems**

CSPs can be classified into different types based on their constraints and problem characteristics:

1. **Binary CSPs**: In these problems, each constraint involves only two variables. For example, in a scheduling problem, the constraint could specify that task A must be completed before task B.

2. **Non-Binary CSPs**: These problems have constraints that involve more than two variables. For instance, in a seating arrangement problem, a constraint could state that three people cannot sit next to each other.

3. **Hard and Soft Constraints**: Hard constraints must be strictly satisfied, while soft constraints can be violated, but at a certain cost. This distinction is often used in real-world applications where not all constraints are equally important.

**Representation of Constraint Satisfaction Problems (CSP)**

In **Constraint Satisfaction Problems (CSP)**, the solution process involves the interaction of variables, domains, and constraints. Below is a structured representation of how CSP is formulated:

1. **Finite Set of Variables** (V1,V2,…,Vn) (*V*1,*V*2,…,*Vn*)**:** The problem consists of a set of variables, each of which needs to be assigned a value that satisfies the given constraints.

2. **Non-Empty Domain for Each Variable** (D1,D2,…,Dn) (*D*1,*D*2,…,*Dn*)**:** Each variable has a domain—a set of possible values that it can take. For example, in a Sudoku puzzle, the domain could be the numbers 1 to 9 for each cell.

3. **Finite Set of Constraints** (C1,C2,…,Cm)(*C*1,*C*2,…,*Cm*)**:** Constraints restrict the possible values that variables can take. Each constraint defines a rule or relationship between variables.

**4. Constraint Representation:**

Each constraint Ci*Ci* is represented as a pair **<scope, relation>**, where:

- **Scope:** The set of variables involved in the constraint.
- **Relation:** A list of valid combinations of variable values thatsatisfy the constraint.

## UNIT II

### REASONING:

- The reasoning is the mental process of deriving logical conclusion andmaking predication from available knowledge, facts, beliefs.
- Reasoning is a way to infer facts from existing data. It is a general process of thinking rationally, to find valid conclusions.
- In AI, the reasoning is essential s0 that the machine can also think rationally as a human brain, and can perform like a human.

### Types of Reasoning:

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning
- Common sense reasoning
- Monotonic reasoning
- Non-monotonic reasoning

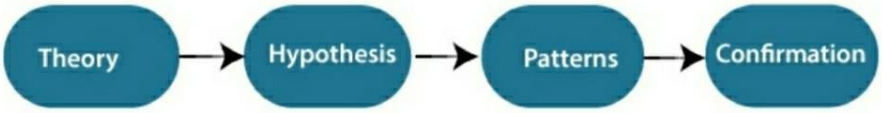
1. **Deductive reasoning**: Deductive reasoning is deducing new. information from logically related known information. It is the form of valid reasoning which means the argument's conclusion must be true when the premises are true.

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

### Example:

Premise-1: All the human eats veggies Premise-2: Suresh is human Conclusion: Suresh eats veggies

General process of deductive reasoning:



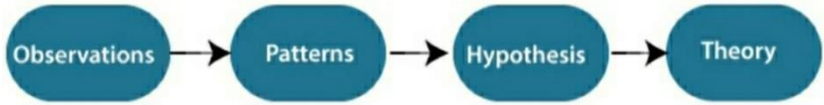

2. **Inductive reasoning**: Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. Series facts It starts with the series of or data and reaches to a general statement or conclusion.

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

In inductive reasoning, premises provide Probable supports to the conclusion, sothe truth of Premises does not guarantee the truth of the conclusion.

Eg: premise: All of the pigeons we have seen in the zoo are white conclusion: Therefore, we can expect all the Pigeons to be white

**General process of Inductive reasoning:**




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. 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 formof reasoning, which can be gained through experiences.

Common sense reasoning simulates the human ability to make presumptionsabout events which occurs on every day

**Example:**

- One person can be at one place at a time

5. **Monotonic Reasoning**: In this 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. I not useful for the real-time system.

In this, adding Knowledge does not decrease the set of prepositions that can bederived.

It is used in conventional reasoning systems, and a logic-based system ismonotonic.

**Example:**

- Earth revolves around the Sun
- It is a true fact, & it cannot be changed even if we add another sentence inknowledge base like, "The moon revolves around the earth" or "Earth is not round", etc.

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

Non-monotonic reasoning deals with incomplete and uncertain models.

**Example:**

- Birds can fly.
- Penguins cannot fly.

- Pitty is a bird.

We conclude pitty can fly. If we add one another sentence in to knowledge base "pitty is a penguin", which concludes "pitty cannot fly", so it invalidates the aboveconclusion.

**SYMBOLIC REASONING:**

- Symbol Artificial Intelligence (AI) is a subfield of AI that focuses on the processing and manipulation of Symbols or concepts, rather than numericaldata.
- Eg: If the patient reports having a fever, the system might use the followingrule.
- IF patient has a fever AND patient has a cough AND patient has difficultybreathing THEN patient may have pneumonia

**Uncertainty:**

- Knowledge representation with first-order logic and propositional logic is based on certainity, means we are sure about the predicates.
- For eg A->B, which means if A is true then B is true, but what about the. Situation cohere we are not sure about whether A is true or not then we cannot express this statement, this situation comes under uncertainity.

**Causes of uncertainity:**

- Information occured from unreliable sources.
- Experimental Errors.
- Equipment fault.
- Temperature Variation.
- Climate change.

**Probabilistic Reasoning:**

- Unpredictable outcomes
- Predicates are too large to handle.
- Unknown error occurs.There are two methods:
- Bayes rule.
- Bayesian statistics.

**1. Bayes rule:**

It determines the probability of an event with uncertain knowledge.Value of P(B|A) with the knowledge of P(A|B)

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

P(A) is called the prior probability. Probability of hypothesis before consideringthe evidence.

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

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

**Example:**

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

The known probability that a patient has meningitis disease is 1/30,000. The known Probability that a patient has a stiff neck is 2%.

P(a|b) = 0.8 P(b) = 1/30000P(a) = 0.2
P(b|a) = P(a|b) P(b) / P(a)

= 0.8[(1/30000)] / 0.02

= 0.001333333

**Application of Bayes' theorem:**

- It is used to calculate the next step of the robot when the already executedstep is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem problem.

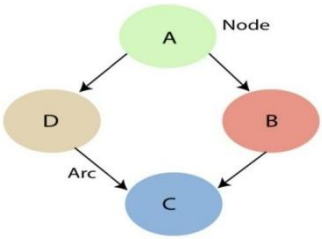
2. **Bayesian Network**:

A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph.

**Bayesian Network consist of two parts:**

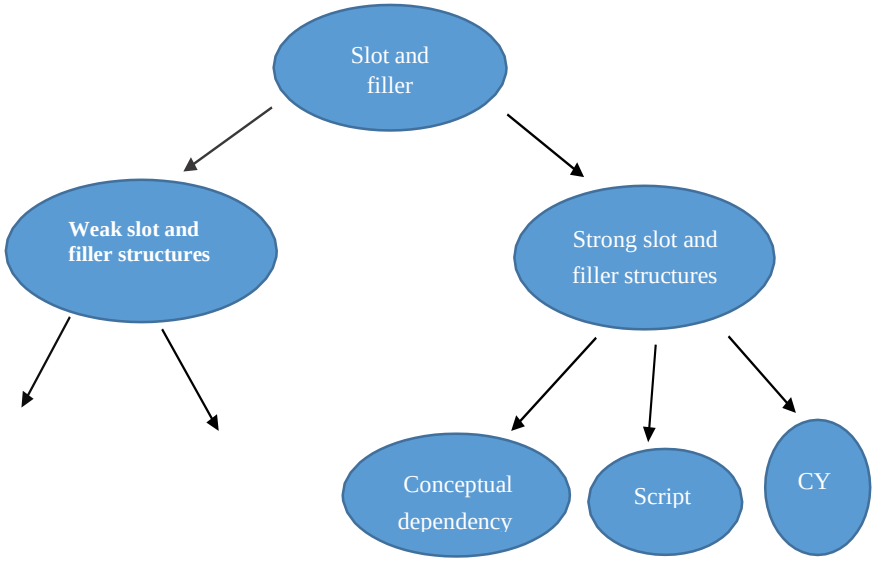
- Directed cyclic Graph.
- Table of Conditional probabilities.

Bayesian network graph is made up of nodes and arcs.




- In above diagram A, B, C, D are random Variables represented by thenodes of the network graph.
- Considering node B, which is connected with node A by a directed arrow,then node A is called the parent of Node B.
- Node C is independent of node A.

## SLOT AND FILLER STRUCTURES:




### Weak slot and filler structures:

"Knowledge-poor" or "weak" as very little importance is given to the specificknowledge the structure should contain.

Attribute slot & its value – filler

1. **Semantic Network**: A Semantic Network (SN) is a simple notation schemefor logical knowledge representation.

A SN consists of a concepts and relations between concepts. Representing a SN with a directed graph:

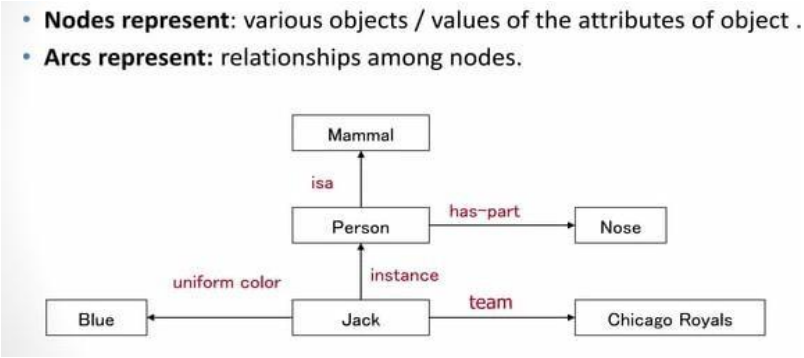
- Vertices: denote concepts
- Edges: represent relation between concepts.

The graphical depiction associated with a SN is significant reason for theirpopularity.

Semantic networks can show natural relationships between objects/concepts.Uses of Semantic Net's:

- Coding static World knowledge
- Built-in fast inference method (inheritance)
- Localization of information

**Example:**




2. **Frames**: Frame is a collection of attributes called as slots and associatedvalues that describe some entity in the world (filler)

It contains information as attribute -value pairs, default values etc. Example:
An example frame corresponding to the semantic net. (Tweety (SPECIES (VALUE bird)) (COLOUR (VALUE Yellow)) (ACTIVITY (VALUE fly))

**Strong slot and filler structures:**

- Conceptual Dependency
- Scripts

- CYC

**Conceptual Dependency:**

- Conceptual Dependency originally developed to represent knowledgeacquired from natural language input.
- CD is collection of Symbols which are used to represent knowledge.
- CD is a theory of how to represent the kind of natural about events that isusually Contained in natural language sentences.

**Various Primitives (Symbols) used in CD:**

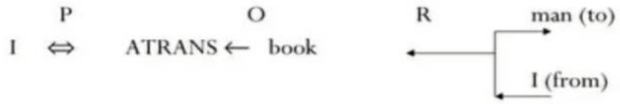
- ATRANS: Transfer of an abstract entity (eg: Give).
- PTRANS: Transfer of an physical Location of an object (eg:go).
- PROPEL: Application of physical force to an Object (eg: push).
- MTRANS: Transfer of mental information (eg: Tell).
- MBUILD: Building a new information out of old. (eg: Decide).
- SPEAK: Utter a sound (eg: Say).
- ATTEND: Focus a sense on a stimulus (eg: Listen, watch).
- INGEST: Actor ingesting an object (eg: eat) .
- MOVE: Movement of a body part by owner (eg: punch, kick).
- GRASP: Actor grasping an object (eg: clutch).
- EXPEL: Actor getting rid of an child object from body ( eg: ????! CRY,Sweat).

**Six primitive Conceptual categories:**

- PP= Real World objects.
- ACT= Real World actions.
- PA= Attributes of objects.
- AA= Attributes of actions
- T= Times

- LOC= Locations

**Example:**




I gave a book to the man.
It should be noted that this representation is same for different saying with samemeaning. For eg:

- I gave the man a book,
- The man got book from me,
- The book was given to man by me etc.

**Scripts:**

- Scripts is a structure that describe a sequence of events in particular context.
- Scripts are frame like structures used to represent commonly occuring experiences such as going to movies, shopping in supermarket, eating in restaurant, banking.
- A Script consist set of slots and information (knowledge) contained in it.

**Various Components of Scripts are:**

- Script Name: Title.
- Track: Special situation, specific variation Roles Peoples involve in the event described in script.

- Entry condition: required pre situation to execute the script.
- Props: non live object involve in the Script.
- Scenes: The actual sequence of events that occur Result Condition that willbe true after events described in the script are occured.

**CYC:**

- An ambitious attempt to form a very large knowledge base aimed atcapturing Commonsense reasoning.
- Initial goals to capture Knowledge from a hundred randomly selected.
- Both Implicit & Explicit knowledge encoded.

Eg., Suppose we read that Wellington Learned of Napoleon's death.

Then we (humans) can conclude Napoleon never new that Wellington had died.We require special implicit knowledge or commonsense such as,

- We only die once.
- You stay dead.
- you cannot learn of anything when dead.
- Time cannot go backward.

## UNIT-III

## KNOWLEDGE REPRESENTATION

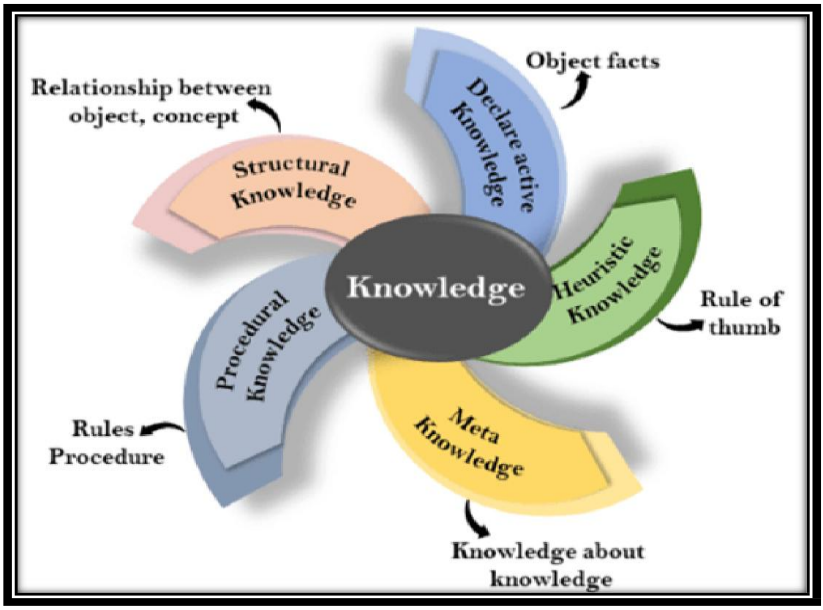
Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**.

**Kinds of knowledge which needs to be represented in AI systems:**

- **OBJECT:** All the facts about objects in our world domain. E.g., Guitarscontains strings, trumpets are brass instruments.
- **EVENTS:** Events are the actions which occur in our world.
- **PERFORMANCE:** It describe behavior which involves knowledge abouthow 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 the 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 artificialintelligence.

## TYPES OF KNOWLEDGE:




## DECLARATIVE KNOWLEDGE:

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

## PROCEDURAL KNOWLEDGE

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible forknowing 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.

**META-KNOWLEDGE:**

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

**HEURISTIC KNOWLEDGE:**

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

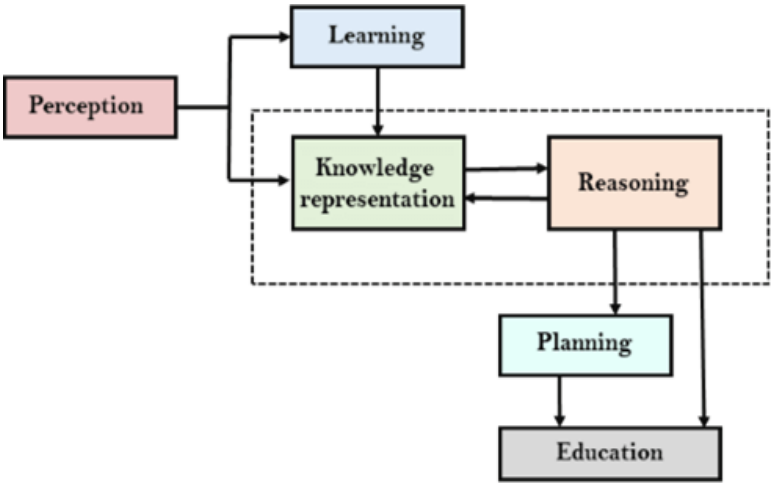
**STRUCTURAL KNOWLEDGE:**

- 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.

**AI KNOWLEDGE CYCLE:**

An Artificial intelligence system has the following components for displayingintelligent behavior:

- Perception
- Learning
- Knowledge Representation and Reasoning
- Planning
- Execution



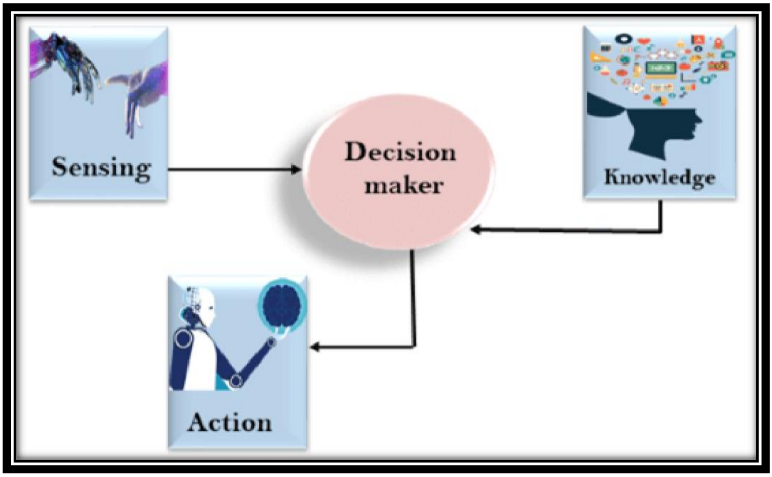

The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, themain components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

## THE RELATION BETWEEN KNOWLEDGE AND INTELLIGENCE:

Knowledge of real-worlds plays a vital role in intelligence and 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 if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.

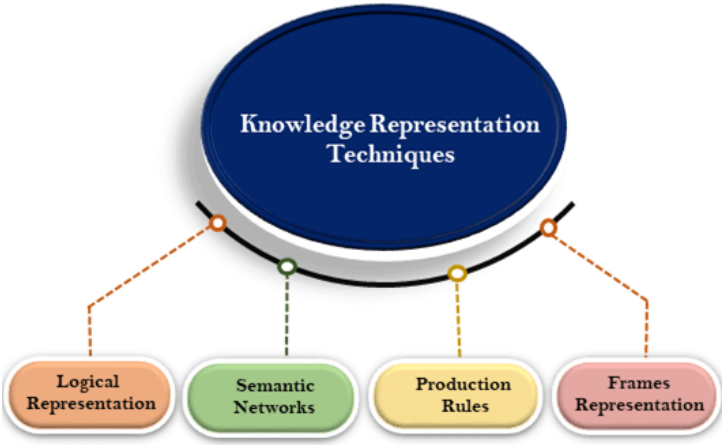
As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.




## TECHNIQUES OF KNOWLEDGE REPRESENTATION

There are mainly four ways of knowledge representation which are given asfollows:

- Logical Representation
- Semantic Network Representation
- Frame Representation
- Production Rules




## 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 intologics using syntax and semantics.

## SYNTAX:

- Syntaxes are the rules which decide how we can construct legal sentences inthe 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:**

- Propositional Logics
- Predicate logics

**Advantages of logical representation:**

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

**Disadvantages of logical Representation:**

- Logical representations have some restrictions and are challenging to workwith.
- Logical representation technique may not be very natural, and inference maynot be so efficient.

**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.

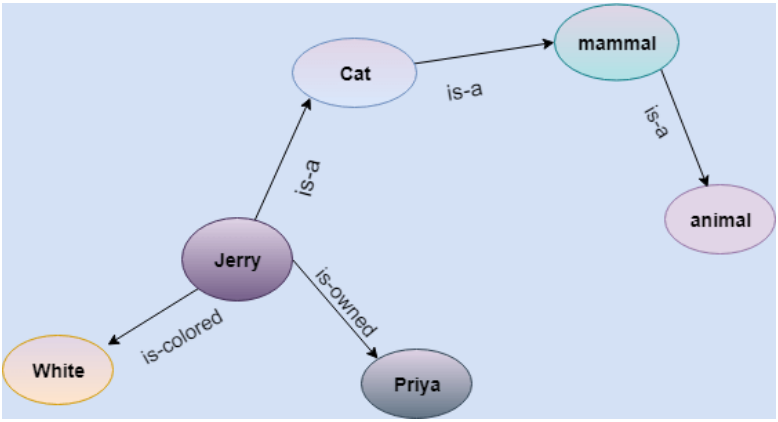
This representation consist of mainly two types of relations:

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

**Example:** Following are some statements which we need to represent in the formof 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.




## DRAWBACKS IN SEMANTIC REPRESENTATION:

- 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.
- 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.

- These types of representations are inadequate as they do not have anyequivalent quantifier, e.g., for all, for some, none, etc.
- Semantic networks do not have any standard definition for the link names.
- These networks are not intelligent and depend on the creator of the system.

**Advantages of Semantic network:**

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

**FRAME REPRESENTATION**

A frame is a record like structure which consists of a collection of attributes and itsvalues to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. Itconsists 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.

**Advantages of frame representation:**

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

**Disadvantages of frame representation:**

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

**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 |

**PRODUCTION RULES**

Production rules system consist of (**condition, action**) pairs which mean, "Ifcondition 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 arule 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 ormodify an individual rule.

**Disadvantages of Production rule:**

1. Production rule system does not exhibit any learning capabilities, as it doesnot 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.

**ISSUES IN KNOWLEDGE REPRESENTATION**

- Important Attributed
- Relationship Among Attributes
- Choosing the Granularity of Representation
- Representing Set of Objects.
- Finding the Right Structures as Needed

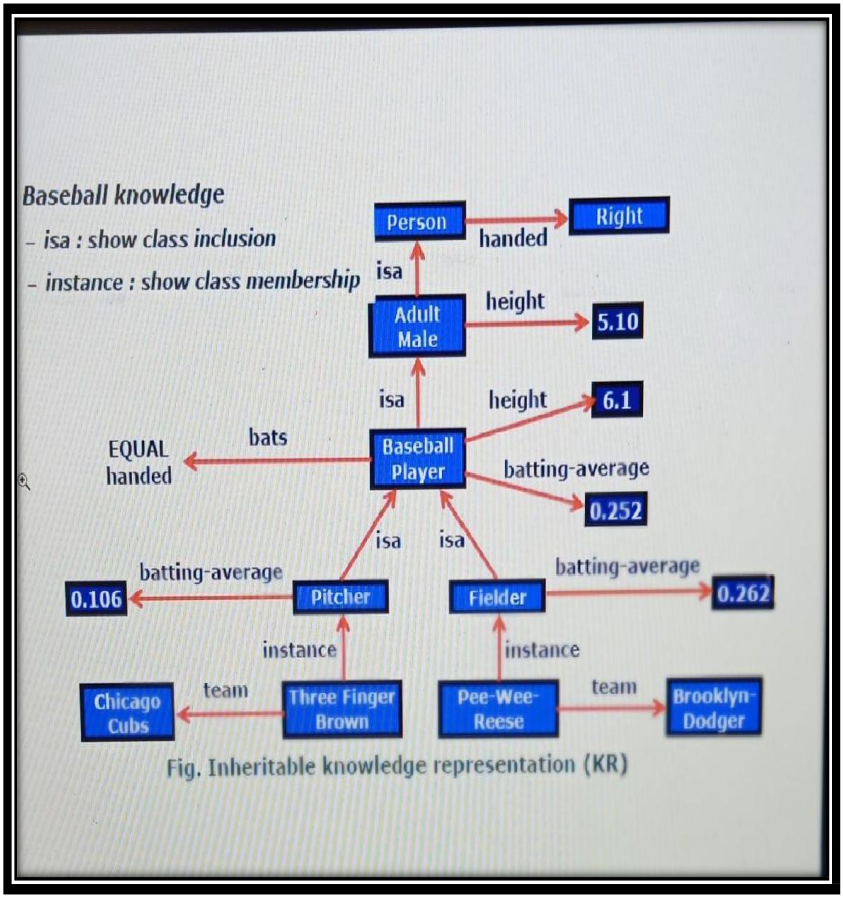



Fig. Inheritable knowledge representation (KR)

## IMPORTANT ATTRIBUTES

- There are attributes that are of general significance
- There are two attributes "instance" and "isa" that are of general importance. These attributes or important because they support property inheritance.

## RELATIONSHIP AMONG ATTRIBUTES

- The attributes to describe objects are themselves entities they represent.
- The relationship between the attributes of an object independent of specific knowledge day in code may hold properties like:

Inverse, existence in an isa hierarchy ,techniques for reasoning about values and single valued attributes.

**Inverse:**

This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways. The figure show attributes (is a, instance and team) each with a directed arrow, originating at the object being described and terminating either at the object or its value.

**There are two ways of realizing this:**

- First, represent two relationships in a single representation; e.g., a logical representation, *team (Pee-Wee-Reese, Brooklyn-Dodgers),* that can be interpreted as a statement about Pee-Wee-Reese or Brooklyn-Dodger.
- Second, use attributes that focus on a single entity but use them in pairs, one the inverse of the other; for e.g., one, *team = Brooklyn-Dodgers*, and the other, *team = Pee-Wee-Reese*,....

This second approach is followed in semantic net and frame-based systems, accompanied by a knowledge acquisition tool that guarantees the consistency of inverse slot by checking, each time a value is added to one attribute then the corresponding value is added to the inverse.

**Existence in an "is a" hierarchy:**

This is about generalization-specialization, like, classes of objects and

specialized subsets of those classes. There are attributes and specialization ofattributes.

Example: the attribute "height" is a specialization of general attribute "physical-size" which is, in turn, a specialization of "physical-attribute".

These generalization-specialization relationships for attributes are Important because they support inheritance.

**Techniques for reasoning about values:**

This is about *reasoning values of attributes* not given explicitly.

Several kinds of information are used in reasoning, like,Height: must be in a unit of length, Age: of person can not be greater than the age of person's parents.

**Single valued attributes:**

This is about a *specific attribute* that is guaranteed to take a unique value.

Example: A baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

**CHOOSING GRANULARITY**

What level should the knowledge be represented and What are the primitives?

- Should there be a small number or should there be a large number oflow-level primitives or High-level facts.
- High-level facts may not be adequate for Inference while Low-levelprimitives may require a lot of storage.

**Example of Granularity:**

- Suppose we are interested in following factsJohn spotted Sue.
- This could be represented as
- Spotted (agent (John), object (Sue))
- Such a representation would make it easy to answer questions suchare
- Who spotted Sue?
- Suppose we want to know

**Did John see Sue?**

- Given only one fact, we cannot discover that answer.
- We can add other facts, such asSpotted (x, y)□ saw (x, y)
- We can now infer the saw (x, y) answer to the question.

**SET OF OBJECTS**

Certain properties of objects that are true as member of a set but not asIndividual;

**Example:** Consider the assertion made in the sentences "there are more sheep than people in Australia", and"English speakers can be found all over the world."

To describe these facts, the only way is to attach assertion to the setsrepresenting people, sheep, and English.

**The reason to represent sets of objects is:**

- If a property is true for all or most elements of a set,
- then it is more efficient to associate it once with the set
- rather than to associate it explicitly with every elements of the set.

**This is done in different ways:**

- In logical representation through the use of universal quantifier, and
- In hierarchical structure where node represent sets, the inheritance *propagate* set level assertion down to individual.

**Example:** assert large (elephant);

Remember to make clear distinction between,

- whether we are asserting some property of the set Itself, means, the set ofelephants is large, (or)
- asserting some property that holds for individual elements of the set,means, any thing that is an elephant is large.

**FINDING RIGHT STRUCTURE**

Access to right structure for describing a particular situation.

It requires, selecting an initial structure and then revising the choice. While doing so, it is necessary to solve following problems:

- how to perform an initial selection of the most appropriate structure.
- how to fill in appropriate details from the current situations.
- how to find a better structure if the one chosen Initially turnsout not to be appropriate.
- what to do if none of the available structures is appropriate.
- when to create and remember a new structure.

There is no good, general purpose method for solving all these problems. Some knowledge representation techniques solve some of them.

## PREDICATE LOGIC

Predicate Logic deals with predicates, which are propositions, consist of variables.

### Predicate Logic - Definition

A predicate is an expression of one or more variables determined on some specific domain. A predicate with variables can be made a proposition by either authorizing a value to the variable or by quantifying the variable.

**The following are some examples of predicates.**

- Consider E(x, y) denote "x = y"
- Consider X(a, b, c) denote "a + b + c = 0"
- Consider M(x, y) denote "x is married to y."

### Quantifier:

The variable of predicates is quantified by quantifiers. There are two types of quantifier in predicate logic - Existential Quantifier and Universal Quantifier.

### Existential Quantifier:

If p(x) is a proposition over the universe U. Then it is denoted as ∃x p(x) and read as "There exists at least one value in the universe of variable x such that p(x) is true. The quantifier ∃ is called the existential quantifier.

There are several ways to write a proposition, with an existential quantifier, i.e.,

(∃x∈A)p(x) or ∃x∈A such that p (x) or (∃x)p(x) or p(x) is true for some x ∈A.

**Universal Quantifier:**

If p(x) is a proposition over the universe U. Then it is denoted as ∀x, p(x) and read as "For every x∈U, p(x) is true." The quantifier ∀ is called the Universal Quantifier.

There are several ways to write a proposition, with a universal quantifier.

∀x∈A, p(x) or p(x), ∀x ∈A Or ∀x, p(x) or p(x) is true for all x ∈A.

**Negation of Quantified Propositions:**

When we negate a quantified proposition, i.e., when a universally quantified proposition is negated, we obtain an existentially quantified proposition, and when an existentially quantified proposition is negated, we obtain a universally quantified proposition.

The two rules for negation of quantified proposition are as follows. These are also called DeMorgan's Law.

**LOGIC SYMBOLS USED IN PREDICATE LOGIC**

| | |
|---|---|
| ∧ | *and* [conjunction] |
| ∨ | *or* [disjunction] |
| ⇒ | *implies* [implication] |
| ¬ | *not* [negation] |
| ∀ | *For all* |
| ∃ | *There exists* |

**First-Order Logic in Artificial Intelligence**

**First-order logic (FOL)**, also known as predicate logic or first-order predicate calculus, is a powerful framework used in various fields such as mathematics, philosophy, linguistics, and computer science. In artificial intelligence (AI), FOL plays a crucial role in knowledge representation, automated reasoning, and naturallanguage processing.

**Fundamentals of First-Order Logic**

1. **Constants**: Constants represent specific objects within the domain of discourse. For example, in a given domain, Alice, 2, and NewYork could be constants.
2. **Variables**: Variables stand for unspecified objects in the domain. Commonlyused symbols for variables include x, y, and z.
3. **Predicates**: Predicates are functions that return true or false, representing properties of objects or relationships between them. For example, Likes(Alice, Bob) indicates that Alice likes Bob, and Greater Than(x, 2) means that x isgreater than 2.
4. **Functions**: Functions map objects to other objects. Forinstance, Mother Of(x) might denote the mother of x.
5. **Quantifiers**: Quantifiers specify the scope of variables. The two mainquantifiers are:
   - **Universal Quantifier (∀)**: Indicates that a predicate applies to all elementsin the domain.
     For example, ∀x (Person(x) → Mortal(x)) means "All persons aremortal."
   - **Existential Quantifier (∃)**: Indicates that there is at least one element in the domain for which the predicate holds.
     For example, ∃x (Person(x) A Likes(x, Ice Cream)) means "Thereexists a person who likes ice cream."

1. **Logical Connectives**: Logical connectives include conjunction (A), disjunction (V), implication (→), biconditional (↔), and negation (¬). These connectives are used to form complex logical statements.

**Example: Using First-Order Logic for Reasoning**

To illustrate the use of FOL in reasoning, consider the following knowledge base:

1. ∀x (Cat(x) → Mammal(x)) (All cats are mammals)
2. ∀x (Mammal(x) → Animal(x)) (All mammals are animals)
3. Cat(Tom) (Tom is a cat).

From these statements, we can infer:

1. Mammal(Tom) (Since Tom is a cat, and all cats are mammals)
2. Animal(Tom) (Since Tom is a mammal, and all mammals are animals).

## KNOWLEDGE REPRESENTATION USING RULES

Knowledge representation using rules in AI refers to the way in which a computer program stores and organizes knowledge in the form of rules. These rules are used to reason, make decisions, and solve problems. Here's adetailed explanation:

### Rule-Based Systems

A rule-based system consists of a set of rules, a knowledge base, and an inference engine. The knowledge base contains a collection of rules, and the inference engine uses these rules to draw conclusions or make decisions.

### Types of Rules

**There are two primary types of rules:**

1. **Forward Chaining Rules**: These rules are used to reason from a set ofpremises to a conclusion. They have the form: "IF A AND B THEN C"

2. **Backward Chaining Rules**: These rules are used to reason from a conclusion to the premises. They have the form: "IF C THEN A AND B"

**Components of a Rule**

**A rule consists of**

1. **Antecedent**: The condition or premise of the rule (A AND B)
2. **Consequent**: The conclusion or action of the rule (C)
3. **Confidence Factor**: A measure of the rule's certainty or reliability (optional)

**How Rules are Used in AI**

Rules are used in various AI applications, including:

1. **Expert Systems**: Mimic the decision-making process of a human expert in aparticular domain
2. **Decision Support Systems**: Provide recommendations or guidance based onrules and data
3. **Natural Language Processing**: Used in parsing and understanding language
4. **Computer Vision**: Used in image recognition and classification

**Advantages of Rule-Based Systems**

1. **Transparency**: Rules provide clear and understandable reasoning
2. **Flexibility:** Easy to add or modify rules as knowledge changes
3. **Explanation Capability**: Can provide explanations for decisions made

**Challenges and Limitations**

1. **Knowledge Acquisition**: Difficult to obtain and encode knowledge in ruleform

2. **Complexity**: Large numbers of rules can lead to complexity and difficulty inmaintenance
3. **Incompleteness**: Rules may not cover all possible situations or exceptions.

## SYNTACTIC REPRESENTATION

Syntactic representation refers to the formal structure or arrangement of symbols and rules used to construct valid statements or expressions in a give language.

### Key aspects of syntactic Representation

- **Symbols**: Basic units of representation.
- (e.g, characters, words, tokens)
- **Grammar:** A set of rules that define the structure of valid expressions.
- **Syntax trees:** Hierarchical structures that represent the syntactic structureof expressions
- **parsing :**The process of analyzing a string of symbols according to thegrammar rules.

### Examples:

(Syntax: $\forall x (P(x) \rightarrow Q(x))$

.

The structure and placement of quantifiers, Predicates, connectives followspecific syntactic rules.

## SEMANTIC REPRESENTATION

Semantic representation refers to the meaning conveyed by syntactic structures. It deals with the interpretation of symbols, expressions and theirrelationships.

**Key aspects of Semantic Representation**

**Meaning:** What the symbols and structures represent.

**Interpretation:** Assigning meaning to syntactic elements.

**Models:** Structures that satisfy the interpretations of the syntactic elements.

**Ontologies**: Formal representations of a set of concepts within a domain andthe relationships between those concepts.

**Examples:**

Semantics: VX (P(x) -> Q(x))

- Here, PLX) might mean "x is a human" and Q (x) might mean "x is mortal."
- The statement means "For all x, if x is a human, then x is mortal."

**LOGIC AND SLOT FILLERS**

- Logic is used to represent knowledge using logical statements,rules and inference techniques.
- It provides a formal way to reason about knowledge, making it possible to draw conclusions, make decisions and solve problems.
- Slot is an attribute Value pair' in its simplest form. Filler is a value that a slotcan take could be a numeric, string Value or a pointer to another slot.
- Slot filler are used to represent knowledge using a frame a-based structure, slots represent attributes or properties
- Fillers represent the values or instances of those attributes

Slot fillers are often used in semantic networks, frames, and ontologies to representKnowledge in a structured and organized way.

**Example:**

**Frame**: Person

**Slots**: name, age, occupation

**Fillers:** John, 30, doctor.

**For eg**: consider a knowledge base that represents information about people,including their name, age, and occupation.

**Logic can be used to represent rules such as**

- "If a person is a doctor, then they have a medical degree."
- "If a person is over 30, then they are Considered experienced.

Slot fillers can be used to represent the attributes and values of individual people,such as:

- John: name John, age = 30, occupation= doctor
- Jane: name: Jane, age= 25, Occupation= engineer.

By combining logic and slot fillers, the Knowledge base can reason about theknowledge and make inferences, such as:

- John has a medical degree because he is a doctor.
- Jane is not considered experienced because she is under 30.

**GAME PLAYING**

Game playing in artificial intelligence refers to the development and application of algorithms that enable computers to engage in strategic decision-making within the context of games. These algorithms, often

termed game playing algorithms in AI, empower machines to mimic human-like gameplay by evaluating potential moves, predicting opponent responses, and making informed choices that lead to favorable outcomes. This concept extends the capabilities of AI systems beyond mere computation and calculation, enabling them to participate in interactive scenarios and make choices based on strategic thinking.

The essence of game playing in AI lies in creating intelligent agents that can navigate the complexities of various games, whether they are traditional board games or digital simulations. The goal is to equip these agents with the ability to assess the current state of the game, anticipate the implications of different moves, and ultimately choose the best course of action. This involves a combination of pattern recognition, probabilistic analysis, and strategic planning, all of which are encapsulated in the game playing algorithm in AI.

**Types of Game Playing in Artificial Intelligence**

Game playing in artificial intelligence encompasses a diverse array of strategies, each aimed at enabling AI systems to excel in games and strategic decision-makingscenarios. These strategies, often referred to as game playing algorithms in AI, can be broadly classified into two main categories: rule-based systems and machine learning-based systems.

**RULE-BASED SYSTEMS**

Rule-based systems, a cornerstone of game playing in AI, rely on predefined sets of rules to govern the behavior of AI agents during gameplay. These rules encapsulate strategies, tactics, and heuristics designed by human experts. These experts analyze the game, anticipate possible moves, and formulate guidelines that the AI adheres to.

In rule-based systems, decisions are based on a deterministic process where each move is evaluated against the predefined rules. These rules dictate how the AI should react to various game states, opponent moves, and potential outcomes. Rule-based approaches are particularly

effective in games with well-defined rules and relatively simple decision trees, such as Tic-Tac-Toe.

## MACHINE LEARNING-BASED SYSTEMS

- Machine learning-based systems, on the other hand, represent a paradigm shift in game playing in AI. Instead of relying on fixed rules, these systems utilize algorithms to learn from experience and adapt their strategies accordingly. These algorithms process vast amounts of data generated through gameplay, identifying patterns, correlations, and optimal moves.

- Reinforcement learning is a prominent example of machine learning-based systems in game playing algorithms in AI. Here, AI agents play games repeatedly, receiving rewards for favorable moves and penalties for unfavorable ones. Over time, the AI learns to maximize rewards by exploring different strategies and refining its decision-making processes. This approach has propelled AI achievements in complex games like Go and chess, showcasing the capacity to tackle intricate decision trees.

## ADVANTAGES OF GAME PLAYING

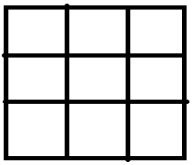
- **Enhanced Strategic Thinking:** Game-playing algorithms empower AI to strategize and make optimal choices, improving decision-making in variousscenarios.
- **Adaptive Learning:** Machine learning-driven approaches enable AI to learn and refine strategies, adapting over time for better performance.
- **Real-world Relevance:** Strategies developed in games find applications in diverse fields, enhancing decision-making in practical situations.
- **Efficient Decision-making:** Algorithms like Alpha-Beta Pruning optimizecomputation, enabling AI to efficiently explore complex

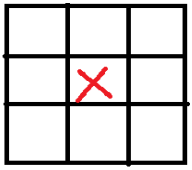
game scenarios.

- **Benchmarking AI Progress:** AI's success in games serves as a marker of advancement, showcasing the growth of AI's strategic capabilities.

## DISADVANTAGES OF GAME PLAYING

- **Computational Complexity:** Game-playing algorithms can be computationally intensive, limiting their real-time application in complexscenarios.
- **Limited Generalization:** Strategies developed for specific games might not readily apply to broader real- world decision-making contexts.
- **Lack of Creativity:** AI's decisions are based on algorithms and past experiences, lacking the creativity and intuition that humans possess.
- **Complexity of Game Rules:** Adapting game-playing algorithms to diverse games with intricate rules can be challenging and time-consuming.
- **Overfitting:** In machine learning-based approaches, there's a risk of overfitting to limited training data, leading to suboptimal decisions in novelsituations.





**Game:** Tic-Tac-Toe

Computer to play a game against a human opponent.

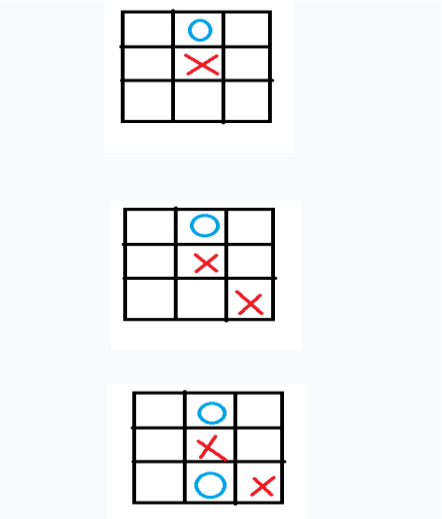
**Objective:** Max aims to win or draw the game by making optimal moves AIprogram

**Game play:**

**Initial State:** The game starts with an empty board.

**Max's Turn:** Max makes the 1st move placing its symbol(x) in position 5.

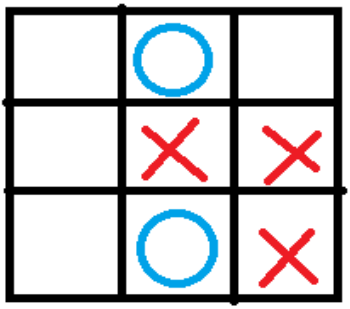
**Human'sTurn:** The human places their symbol (0) in position 2



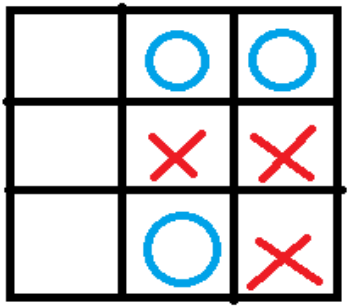
**Max's Turn:** Max analyzes the board and decides to place its symbol(x) inposition 9

**Human Turn:** The human places their symbol(0) in position 8.

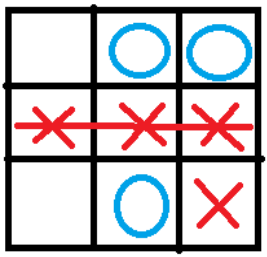
**Max's Turn:** Max analyzes the board again and decides to place its symbol(x) in position 6.



**Human Turn:** The human places their symbol(0) in position 3.



**Max's Turn:** Max analyses the board and decicdes to place its symbol(x) in position 4.



**GAME OVER: MAX WINS THE GAME!**

**MINIMAL SEARCH**

Minimax search is a decision-making algorithm used in Artificial Intelligence (AI) to find the best move in a game or decision-making situation. It's commonly used in games like chess, tic-tac-toe, and Go, but also has applications in other fields like economics and resource allocation.

### How Minimax Search Works

1. Game Tree: The algorithm starts by building a game tree, which represents all possible moves and their outcomes.
2. Node Evaluation: Each node in the tree is evaluated using a heuristic function, which estimates the node's value.
3. Minimax: The algorithm applies the minimax principle, which states that the best move is the one that minimizes the maximum potential loss (or maximizes the minimum potential gain).
4. Recursive Search: Minimax search recursively explores the game tree, considering all possible moves and their outcomes.
5. Alpha-Beta Pruning: To optimize the search, alpha-beta pruning is used to eliminate branches that will not affect the final decision.

### Key Concepts

- **Maximizer**: The player who wants to maximize their score (usually the AI).
- **Minimizer**: The player who wants to minimize their score (usually theopponent).
- **Ply**: A single move in the game.
- **Depth:** The number of plies explored in the search.
- **Heuristic Function**: Estimates the value of a node.

### Minimax Search Algorithm

1. Initialize the game tree and set the current node to the root.
2. Evaluate the current node using the heuristic function.
3. If the current node is a leaf node, return its value.
4. If the current node is a maximizer node:
    - For each child node, recursively call the minimax function.
    - Return the maximum value of the child nodes.
5. If the current node is a minimizer node:
    - For each child node, recursively call the minimax function.
    - Return the minimum value of the child nodes.
6. Apply alpha-beta pruning to eliminate branches.
7. Return the best move (the one that maximizes the minimum potential gain).

### Advantages and DisadvantagesAdvantages:

- Finds the optimal move in a game or decision-making situation.
- Can handle complex games with many possible moves.

**Disadvantages:**

- Can be computationally expensive.
- May get stuck in an infinite loop if the game tree is too deep.

**Real-World Applications**

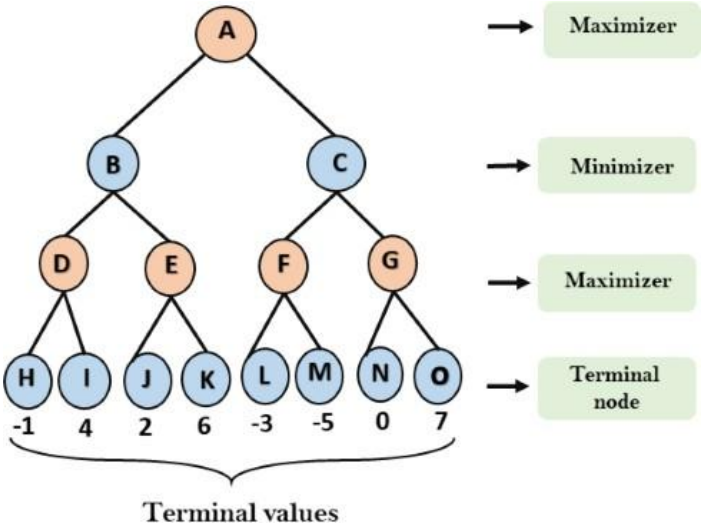
- Game playing (chess, Go, tic-tac-toe, etc.)
- Resource allocation
- Economics
- Decision-making under uncertainty

**EXAMPLE FOR MINMAX SEARCH**

**Working of Min-Max Algorithm:**

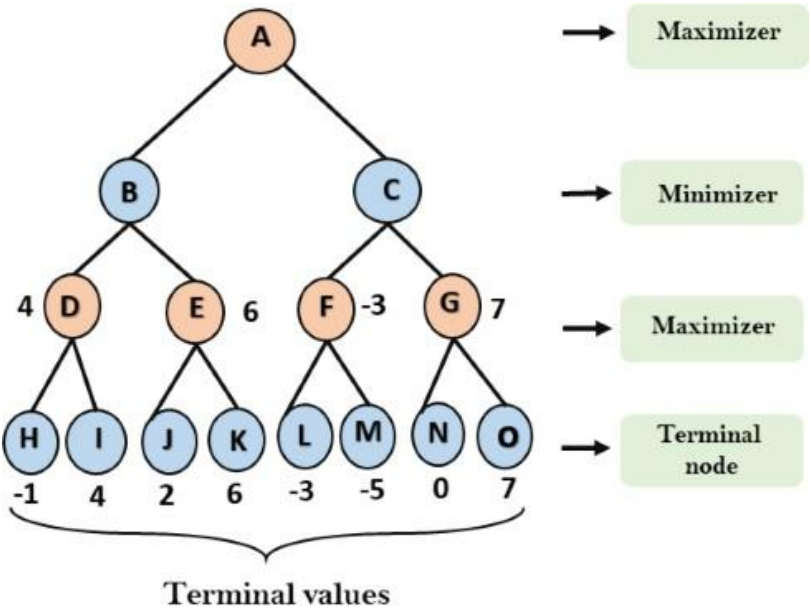
- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

**Step-1:** In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



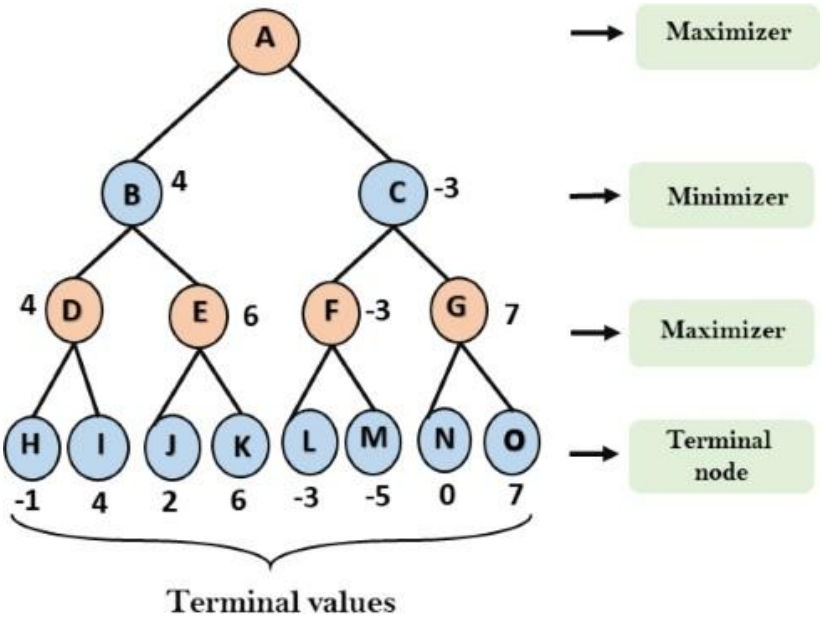

**Step 2:** Now, first we find the utilities value for the Maximizer, its initial value is - ∞, so we will compare each value in terminal state with initial value of Maximizerand determines the higher nodes values. It will find the maximum among the all.

- For node D max(-1,- -∞) => max(-1,4)= 4
- For Node E max(2, -∞) => max(2, 6)= 6
- For Node F max(-3, -∞) => max(-3,-5) = -3
- For node G max(0, -∞) = max(0, 7) = 7



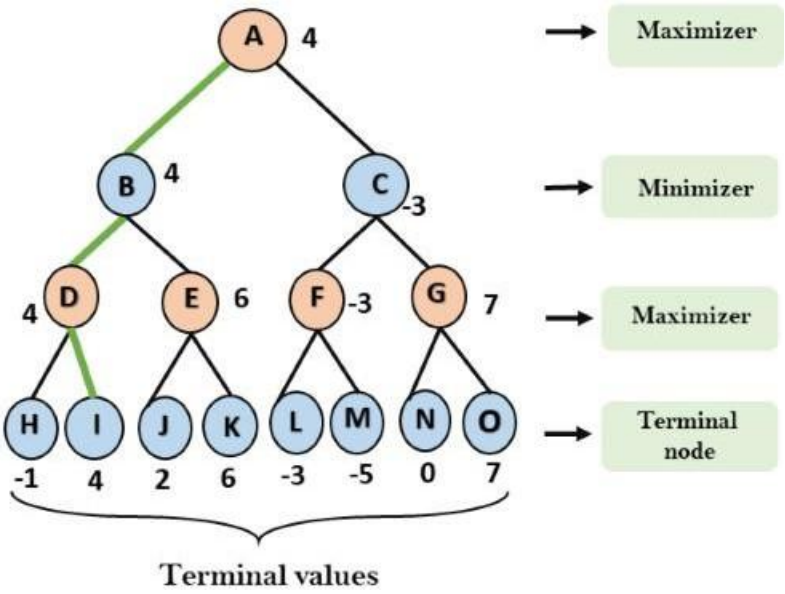

**Step 3:** In the next step, it's a turn for minimizer, so it will compare all nodes valuewith +∞, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3




**Step 4:** Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- For node A max(4, -3)= 4




That was the complete workflow of the minimax two player game.

**Properties of Mini-Max algorithm:**

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution(if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similarto DFS which is **O(bm)**.

## ALPHA- BETA CUTOFF (OR) PRUNING

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- **The two-parameter can be defined as:**

  **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.

  **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

### Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is: $\alpha >= \beta$

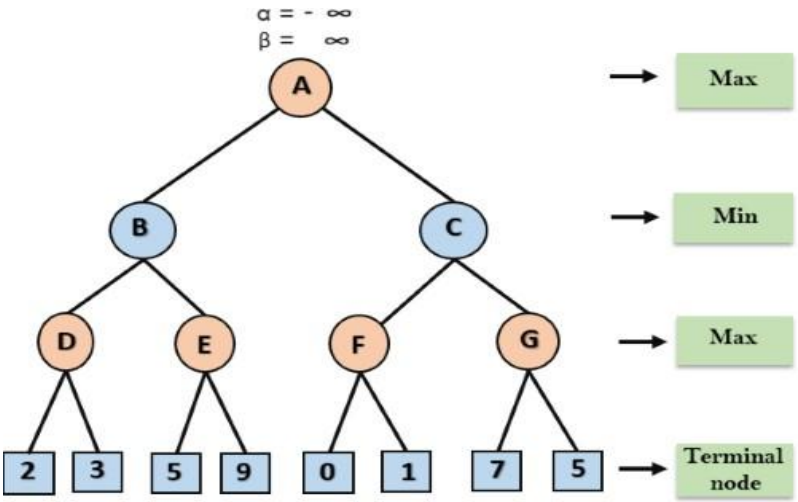
**Key points about alpha-beta pruning:**

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

**Working of Alpha-Beta Pruning:**

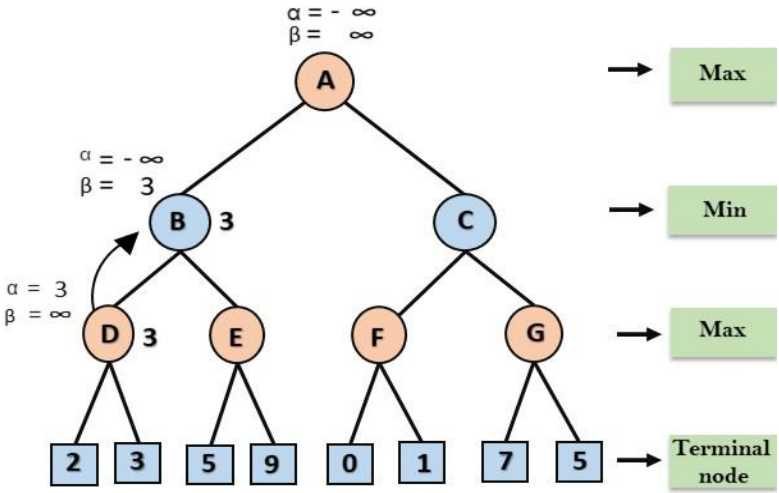
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

**Step 1:** At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



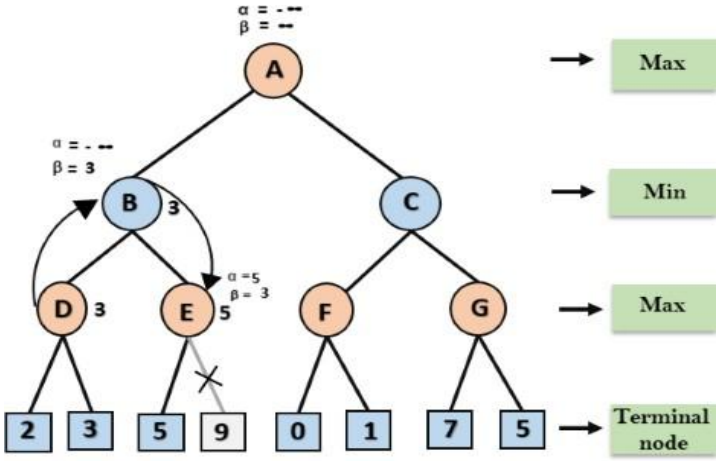
**Step 2:** At Node D, the value of α will be calculated as its turn for Max. The valueof α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the valueof α at node D and node value will also 3.

**Step 3:** Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.




In the next step, algorithm traverse the next successor of Node B which is node E,and the values of α= -∞, and β= 3 will also be passed.

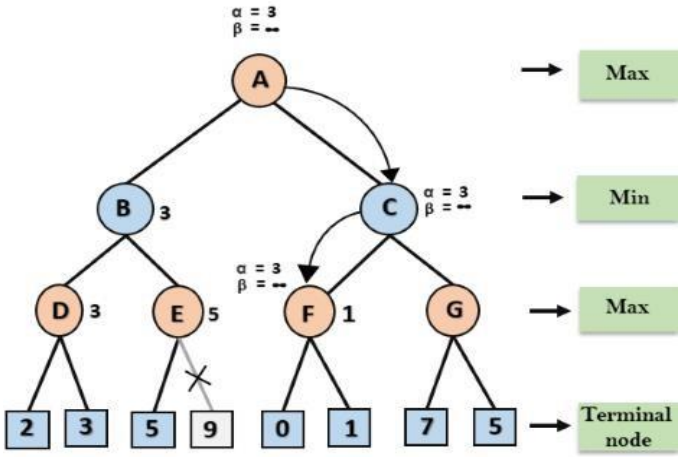
**Step 4:** At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.




**Step 5:** At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 asmax (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

At node C, α=3 and β= +∞, and the same values will be passed on to node F.

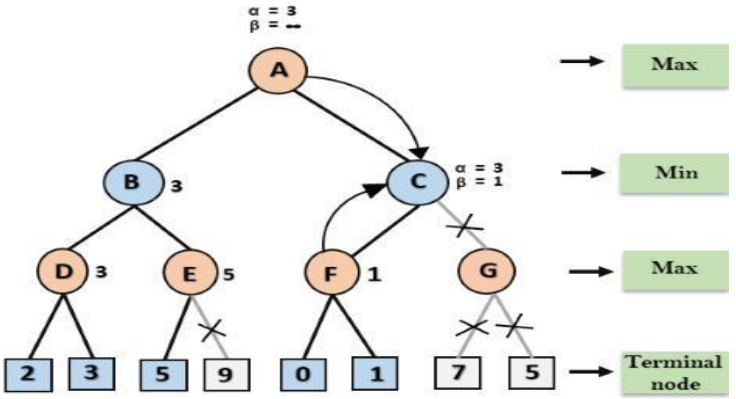
**Step 6:** At node F, again the value of α will be compared with left child which is 0,and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.

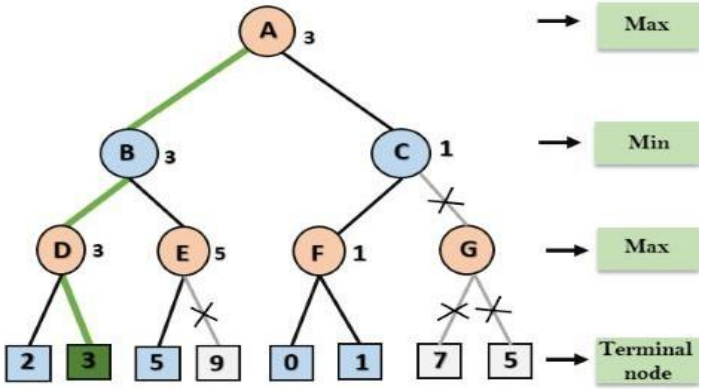



**Step 7:** Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.

**Step 8:** C now returns the value of 1 to A here the best value for A is max (3, 1) =3

Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for themaximizer is 3 for this example.


α = 3
β = ∞
A
B 3
C
α = 3
β = 1
D 3
E 5
F 1
G
2
3
5
9
0
1
7
5
Max
Min
Max
Terminal node


A 3
B 3
C 1
D 3
E 5
F 1
G
2
3
5
9
0
1
7
5
Max
Min
Max
Terminal node

## ITERATIVE DEEPENING PLANNING

Iterative deepening planning is a planning strategy that combines the benefits of iterative deepening search and planning. It is used to find a plan that achieves a goal by iteratively deepening the search for a solution.

**Here's a step-by-step explanation of iterative deepening planning:**

- Initialize the planning problem, including the goal, initial state, and actions.
- Set the initial depth limit to 1.
- Perform a breadth-first search (BFS) or depth-first search(DFS) up to the current depth limit.
- If a solution is found, return the plan.
- If the search reaches the depth limit without finding a solution, increment the depth limit by 1.
- Repeat steps 3-5 until a solution is found or the maximum depth limit is reached.

**Iterative deepening planning combines the benefits of:**

- Iterative deepening search: efficiently explores the search space bygradually increasing the depth limit.
- Planning: generates a sequence of actions to achieve the goal.

**Advantages:**

- Finds a plan that achieves the goal.
- Efficiently explores the search space.
- Can handle complex planning problems.

**Disadvantages:**

- May get stuck in an infinite loop if the planning problem has no solution.

- Can be computationally expensive for very complex problems.

**Real-world applications:**

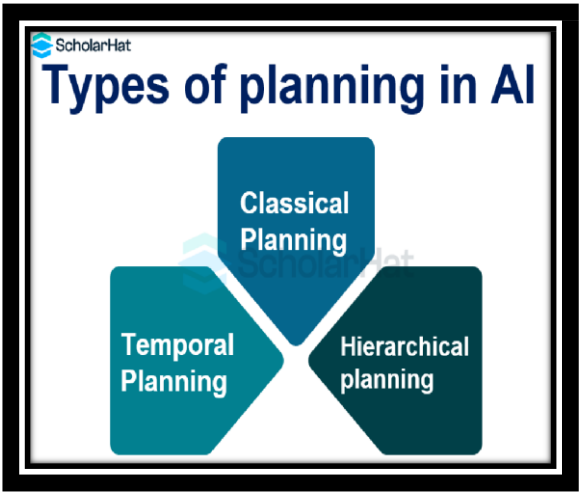
- Robotics: motion planning, task planning
- Logistics: planning routes, scheduling deliveries
- Manufacturing: planning production, scheduling maintenance.

## COMPONENT OF PLANNING SYSTEM

### What is Planning in AI?

Planning in AI is the process of coming up with a series of actions or procedures to accomplish a particular goal. It entails assessing the existing situation, identifying the intended outcome, and developing a strategy that specifies the steps to take to get there. It is not confined to a particular industry; it may also be used in robots, video games, logistics, as well as healthcare.

### Different types of planning in AI



In planning comes in different types, each suitable for a particular situation.Popular different types of planning in ai include:

- **Classical Planning:** In this style of planning, a series of actions is created to accomplish a goal in a predetermined setting. It assumes that everything is static and predictable.
- **Hierarchical planning:** By dividing large problems into smaller ones, hierarchical planning makes planning more effective. A hierarchy of plans must be established, with higher-level plans supervising the execution of lower-level plans.
- **Temporal Planning:** Planning for the future considers time restrictions and interdependencies between actions. It ensures that the plan is workable within a certain time limit by taking into account the duration of tasks.

**Components of planning system in AI**

A planning system in AI is made up of many crucial parts that cooperate to produce successful plans. These components of planning system in ai consist of:

- **Representation:** The component that describes how the planning problem is represented is called representation. The state space, actions, objectives, and limitations must all be defined.
- **Search:** To locate a series of steps that will get you where you want to go, the search component searches the state space. To locate the best plans, a variety of search techniques, including depth-first search & A* search, canbe used.
- **Heuristics:** Heuristics are used to direct search efforts and gauge the expense or benefit of certain actions. They aid in locating prospective routes and enhancing the effectiveness of the planning process.

**Benefits of AI Planning**

Numerous advantages of AI planning contribute to the efficacy and efficiency ofartificial intelligence systems. Some key benefits include:

- **Resource Allocation:** With the help of AI planning, resources can be distributed in the best way possible, ensuring that they are used effectively toaccomplish the desired objectives.
- **Better Decision-Making:** AI planning aids in making knowledgeable judgments by taking a variety of aspects and restrictions into account. It helps AI systems to weigh several possibilities and decide on the best courseof action.
- **Automation of Complex Tasks:** AI planning automates complicated tasks that would otherwise need a lot of human work. It makes it possible for AI systems to manage complex procedures and optimize them for better results.

**Applications of AI Planning**

AI planning is used in many different fields, demonstrating its adaptability andefficiency. A few significant applications are:

- **Robotics:** To enable autonomous robots to properly navigate their surroundings, carry out activities, and achieve goals, planning is crucial.
- **Gaming:** AI planning is essential to the gaming industry because it enables game characters to make thoughtful choices and design difficult and interesting gameplay scenarios.
- **Logistics:** To optimize routes, timetables, and resource allocation and achieve effective supply chain management, AI planning is widely utilized in logistics.
- **Healthcare:** AI planning is used in the industry to better the quality and effectiveness of healthcare services by scheduling patients, allocating resources, and planning treatments.

**Challenges in AI Planning**

While AI planning has many advantages, many issues need to be resolved. Typicalchallenges include:

- **Complexity:** Due to the wide state space, multiple possible actions, and interdependencies between them, planning in complicated domains can be difficult.
- **Uncertainty:** One of the biggest challenges in AI planning is overcoming uncertainty. Actions' results might not always be anticipated, thus the planning system needs to be able to deal with such ambiguous situations.
- **Scalability:** Scalability becomes a significant barrier as the complexity and scale of planning problems rise. Large-scale issues must be effectively handled via planning systems.

**Strategies for Mastering AI Planning**

Adopting strategies that improve planning abilities is crucial if you want to masterAI planning. Here are some strategies to think about:

- **Domain Knowledge:** Learn everything there is to know about the planning domain. Better strategies can be made if you are aware of the complexities and limitations of the domain.
- **Algorithm Selection:** It is essential to choose the right planning algorithm for the particular issue at hand. Choosing the best algorithm can have a big impact on the planning process because different algorithms have different strengths and disadvantages.
- **Improvement through iteration:** Planning is an iterative process, and improvement is essential. Analyze the effectiveness of plans, pinpoint areas that need improvement, and adjust the planning system as necessary.

### Tools and Techniques for AI Planning

Various tools and strategies that support planning can be used to facilitate AI planning. Techniques and tools that are commonly used include:

- **Automated planners:** Programmes that generate plans automatically include STRIPS and PDDL, which offer a framework for specifying planning problems.
- **Constraint Programming:** Using the strong technique of constraint programming, complicated planning issues with a variety of constraints can be modeled and solved.
- **Machine Learning:** Reinforcement learning is a machine learning technique that can be used to enhance planning by learning from previous experiences and refining plans in response to feedback.

## GOAL STACK PLANNING

### What is Goal Stack Planning?

Goal Stack Planning is one of the earliest methods in artificial intelligence in whichwe work **backwards from the goal state to the initial state.**

We start at the goal state and we try fulfilling the preconditions required to achievethe initial state. These preconditions in turn have their own set of preconditions, which are required to be satisfied first. We keep solving these “goals” and “sub- goals” until we finally arrive at the Initial State. **We make use of a stack to hold these goals that need to be fulfilled as well the actions that we need to performfor the same**.

Apart from the “Initial State” and the “Goal State”, we maintain a **“World State”** configuration as well. Goal Stack uses this world state to work its way fromGoal State to Initial State. World State on the other hand starts off as the Initial State and ends up being transformed into the Goal state.

At the end of this algorithm we are left with an empty stack and a set of

actionswhich helps us navigate from the Initial State to the World State.

**Representing the configurations as a list of "predicates"**

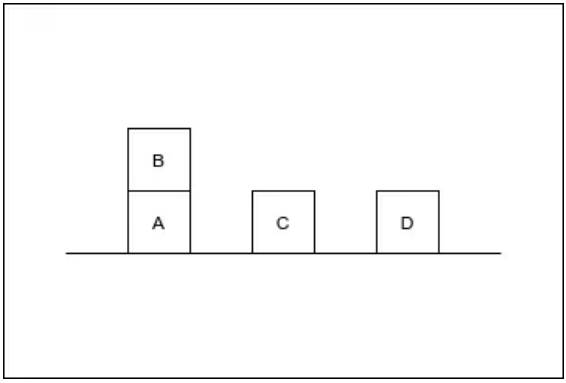
Predicates can be thought of as a statement which helps us convey the informationabout a configuration in Blocks World.

Given below are the list of predicates as well as their intended meaning

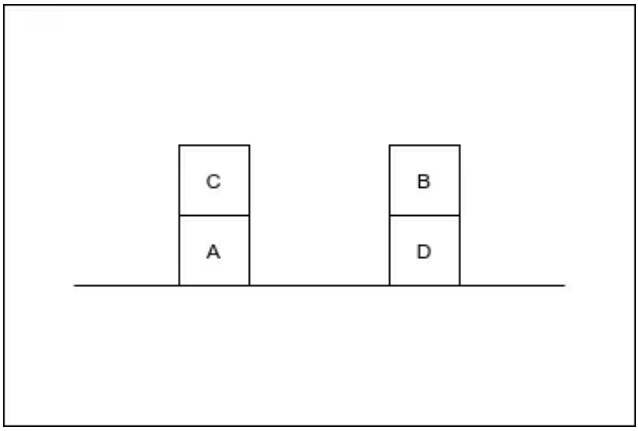
1. ON(A,B) : Block A is on B
2. ONTABLE(A) : A is on table
3. CLEAR(A) : Nothing is on top of A
4. HOLDING(A) : Arm is holding A.
5. ARMEMPTY : Arm is holding nothing

Using these predicates, we represent the Initial State and the Goal State in ourexample like this:

**Initial State** — ON(B,A) A ONTABLE(A) A ONTABLE(C) A ONTABLE(D) A CLEAR(B) A CLEAR(C) A CLEAR(D) A ARMEMPTY




**Goal State** — ON(C,A) A ON(B,D) A ONTABLE(A) A ONTABLE(D) A CLEAR(B) A CLEAR(C) A ARMEMPTY




Thus a configuration can be thought of as a list of predicates describing the currentscenario.

**"Operations" performed by the robot arm**

The Robot Arm can perform 4 operations:

1. STACK(X,Y) : Stacking Block X on Block Y
2. UNSTACK(X,Y) : Picking up Block X which is on top of Block Y
3. PICKUP(X) : Picking up Block X which is on top of the table
4. PUTDOWN(X) : Put Block X on the table

All the four operations have certain preconditions which need to be satisfied to perform the same. These preconditions are represented in the form of predicates.

The effect of these operations is represented using two lists **ADD** and **DELETE**. DELETE List contains the predicates which will cease to be true once the operationis performed. ADD List on the other hand contains the predicates which will become true once the operation is performed.

The Precondition, Add and Delete List for each operation is rather intuitive andhave been listed below.

| OPERATORS | PRECONDITION | DELETE | ADD |
|---|---|---|---|
| STACK(X,Y) | CLEAR(Y)∧ HOLDING(X) | CLEAR(Y) HOLDING(X) | ARMEMPTY ON(X,Y) |
| UNSTACK(X,Y) | ARMEMPTY∧ ON(X,Y)∧ CLEAR(X) | ARMEMPTY∧ ON(X,Y) | HOLDING(X) ∧CLEAR(Y) |
| PICKUP(X) | CLEAR(X)∧ ONTABLE(X)∧ ARMEMPTY | ONTABLE(X)∧ ARMEMPTY | HOLDING(X) |
| PUTDOWN(X) | HOLDING(X) | HOLDING(X) | ONTABLE(X)∧ ARMEMPTY |

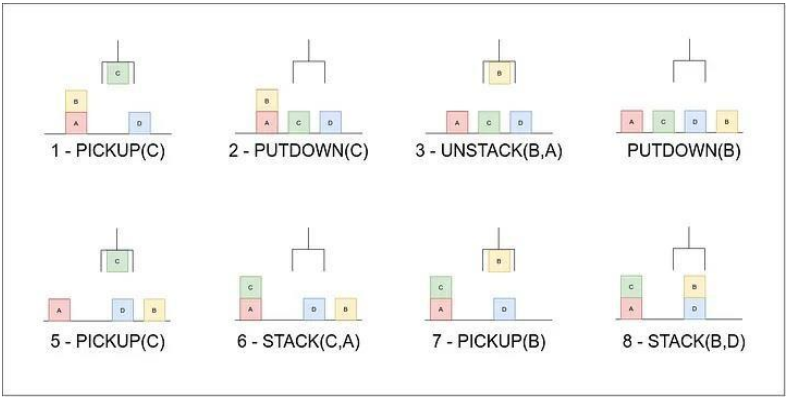
For example, to perform the **STACK(X,Y)** operation i.e. to Stack Block X on top of Block Y, No other block should be on top of Y **(CLEAR(Y))** and the Robot Armshould be holding the Block X (**HOLDING(X)**).

Once the operation is performed, these predicates will cease to be true, thus they are included in **DELETE List** as well. (Note : It is not necessary for the Precondition and DELETE List to be the exact same).

On the other hand, once the operation is performed, The robot arm will be free(**ARMEMPTY**) and the block X will be on top of Y (**ON(X,Y)**).

The other 3 Operators follow similar logic, and this part is the cornerstone of Goal Stack Planning. In this example, steps = [PICKUP(C), PUTDOWN(C), UNSTACK(B,A), PUTDOWN(B), PICKUP(C), STACK(C,A), PICKUP(B), STACK(B,D)]

The visual representation of our steps variable looks like this.



## UNIT-IV

## NATURAL LANGUAGE PROCESSING (NLP)

### NLP:

NLP is a field of computer science, artificial intelligence and linguistics that studies how humans and computers interact with language.

### NLP USED FOR:

NLP is a machine learning technology that gives computers the ability to interpt, manipulate, and comprehend human language.

### IS NLP AI or ML?

NLP is a subfield of computer science and artificial intelligence(AI) that uses machine learning to enable computers to understand and communicate with humanlanguage.

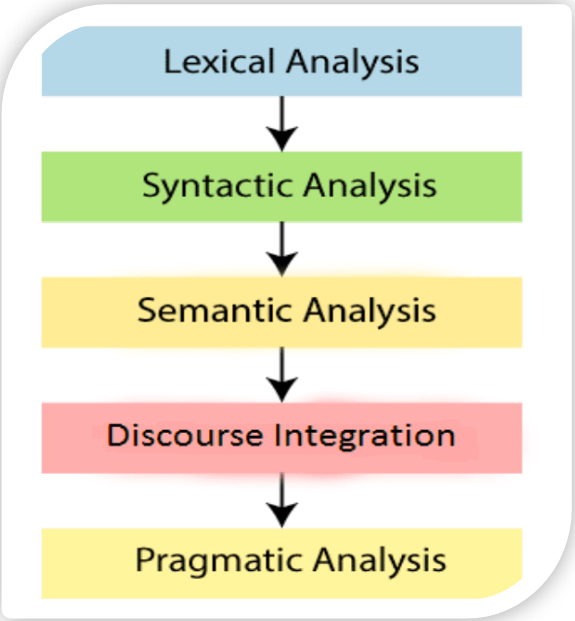
### IS NLP THE FUTURE OF AI?

NLP stands as a advancing domain with extensive applications across diverse industrial sectors. Its surge in popularity within these sectors can be attributed to the exponential growth of AI driven technology.

### IS NLP A PROGRAMMING LANGUAGE?

It is an ontology assisted way of programming in terms of natural language sentences.

**LAYERS OF NATURAL LANGUAGE PROCESSING(NLP).**




**LEXICAL ANALYSIS:**

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

**SYNTACTIC ANALYSIS:**

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

**SEMANTIC ANALYSIS:**

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

**DISCOURSE INTEGRATION:**

- Discourse integration depends upon the sentences that proceeds it and also invokes the meaning of the sentencethat follow it.

**PRAGMATIC ANALYSIS:**

- It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

**ADVANTAGES OF NLP:**

- NLP helps users to ask questions about any subject and get a direct response within seconds.
- NLP helps computers to communicate with humans in their language.
- It is very time efficient.

**DISADVANTAGES OF NLP:**

- NLP is unpredictable.
- **NLP may not show context**
- NLP may require more keystrokes.

**NLP TECHNIQUES:**

NLP encompasses a wide array of technique that aimed at enabling computers to process and understand human language.

- Test processing and preprocessing in NLP
  - Dividing text into smaller units, such as words or sentences
  - Reducing words to their base or root forms.
  - Removing common words(like “and” , “the” , “is”) that may notcarry significant meaning.

- Syntax and parsing in NLP
  - Assigning parts of speech to each word in a sentence(eg.,noun, verb, adjective)
  - Analyzing the grammatical structure of a sentence to identifyrelationships between words.
  - Breaking down a sentence into its constituent parts or phrases(eg.,noun phrases, verb phrases).

- Semantic Analysis
  - Identifying and classifying entities in text, such as names of people,organizations, locations, dates, etc.,
  - Determining which meaning of a word is used in a given context.
  - Identifying when different words refer to the same entity in atext.(eg., “he” refers to “John”).

- Text classification in NLP
  - Identifying topics or themes within a large collection of documents.
  - Classifying text as spam or not spam.

- Question Answering
    - Retrival based QA: Finding and returning the most relevant textpassage in response to a query.
    - Generating an answer based on the information available in a textcorpus.

**FUTURE SCOPE OF NLP**

- NLP has a promising future and is expected to improve existing technologies and make interactions with technology more natural.
- It has numerous possibilities and applications. Advancements in field like sppech recognition, automated machine translation, sentiment analysis and chatbots.

**FUTURE ENHANCEMENTS OF NLP**

- The future of NLP holds exciting possibilities across various sectors.
- Healthcare: NLP can transform patient care through improved diagnostics, personalized treatment plans and efficient patient doctor communication.

**SYNTACTIC PROCESS IN NLP**

Syntactic processing is the process of analyzing the grammatical structure ofa sentence to understand its meaning.

This involves identifying the different parts of speech in a sentence, such as nouns, verbs, adjectives, adverbs and how they relate to each other in order to give proper meaning to the sentence.

**SYNTACTIC PROCESSING WORK**

“The cat sat on the mat” “cat” as a noun “sat” as a verb “on” as a preposition“mat” as a noun

It would also involve understanding that “cat” is the subject of the sentence and“mat” is the object.

## APPLICATIONS OF SYNTACTIC PROCESSING

1. **Language translation**:

Understanding the syntactic structure of a sentence is crucial foraccurate translation.

2. **Sentiment analysis:**

Identifying the relationships between words and phrases helps determine the sentiment of a text.

3. **Question answering:**

Syntactic processing helps identify the relationships between entities and actions in a text.

4. **Text summarization:**

Understanding the syntactic structure of a text helps identify the most important information.

## SEMANTIC ANALYSIS IN NLP

Semantic analysis is a technique used in NLP to help machines understand the meaning of words, sentences and texts by considering the context ofthe text.

## EXAMPLE:

“The boy ate the apple” defines an apple as a fruit. while, “The boy went to Apple” define Apple as a brand or store.

## APPLICATIONS OF SEMANTIC ANALYSIS

### 1. Information Retrieval:

It improves search engines by understanding the meaning behind user queries and document content.

### 2. Question Answering:

Semantic analysis helps answer complex questions by identifying relevant information and relationships in text.

### 3. Text Summarization:

Semantic analysis summarizes long documents by extracting key concepts and relationships.

### 4. Sentiment Analysis:

It determines the sentiment and emotional tone behind text, such as detecting positive or negative opinions.

### 5. Named Entity Recognition:

Semantic analysis identifies and categorizes named entities (people, places, organizations) in text.

### 6. Relationship Extraction:

Semantic analysis identifies relationships between entities, such as "person A is a colleague of person B".

## ADVANTAGES OF SEMANTIC ANALYSIS

### 1. Improved understanding:

Semantic analysis provides a deeper understanding of themeaning and context of text, leading to more accurate interpretations.

**2. Enhanced information retrieval:**

Semantic analysis improves search results by capturing the nuances of language and returning more relevant information.

**3. Better sentiment analysis:**

Semantic analysis accurately identifies sentiment and emotional tone, helping businesses and organizations understand public opinion.

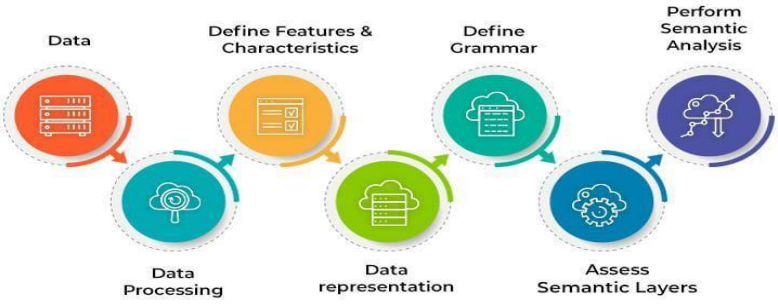
**4. More accurate entity recognition:**

Semantic analysis identifies and categorizes named entities, improving data quality and facilitating data integration.

**5. Increased accuracy in question answering:**

It provides more accurate answers to complex questions by understanding relationships and context.

**SEMANTIC ANALYSIS WORKS**



## ELEMENTS OF SEMANTIC ANALYSIS

**Hyponyms:**

This refers to a specific lexical entity having a relationship with a more generic verbal entity called hypernym.

**For example**: red, blue, and green are all hyponyms of color, their hypernym.

**Meronomy:**

Refers to the arrangement of words and text that denote a minor component of something.

**For example**: mango is a meronomy of a mango tree.

**Polysemy:**

It refers to a word having more than one meaning. However, it is represented under one entry.

**For example**: the term 'dish' is a noun. In the sentence, 'arrange the disheson the shelf,' the word dishes refers to a kind of plate.

**Synonyms:**

This refers to similar-meaning words.

**For example**: abstract (noun) has a synonyms summary–synopsis.

**Antonyms:**

This refers to words with opposite meanings.

**For example:** cold has the antonyms warm and hot.

**Homonyms:**

This refers to words with the same spelling andpronunciation, but reveal a different meaning altogether.

**For example:** bark (tree) and bark (dog).

## TASKS INVOLVED IN SEMANTIC ANALYSIS

- Word sense Disambiguation
- Relationship extraction

## WORD SENSE DISAMBIGUATION:

It refers to an automated process of determining the sense or meaning of the word in a given context.

## EXAMPLE:

'Raspberry Pi' can refer to a fruit, a single-board computer, or even a company (UK-based foundation). Hence, it is critical to identify which meaning suits the word depending on its usage.

## RELATIONSHIP EXTRACTION:

It determine the semantic relationship between words in a text. In this, relationship include various entities such as an individuals name,place, company, designation, etc.,

## EXAMPLE:

Elon Musk is one of the co-founders of Tesla, which is based in Austin, Texas.

- This phrase illustrates two different relationships.
- Elon Musk is the co-founder of Tesla[Person] [Company]
- Tesla is based in Austin, Texas[Company] [Place]

**PARALLEL AND DISTRIBUTED AI PSYCOLOGICAL MODELLING**

It refers to the use of parallel and distributed computing techniques to model human cognition and behavior. This involves.

**1. Parallel processing:**

It using multiple processing units to perform tasks simultaneously, mimicking the brain's parallel processing capabilities.

**2. Distributed processing:**

It breaking down complex tasks into smaller sub-tasks and distributing them across multiple processing units or agents, similar tohow cognitive tasks are distributed across different brain regions.

**GOALS:**

**Scalability:**

Model complex cognitive phenomena that require large-scaleprocessing.

**Flexibility:**

Accommodate individual differences and adapt to changing environments.

**Real-time processing:**

Enable real-time interaction and feedback.

**APPLICATIONS:**

**Cognitive architectures:**

Integrate parallel and distributed processing into cognitive models.

**Neural networks:**

Use parallel and distributed computing to train and deploy neural networks.

**Multi-agent systems:**

Model social behavior and interactions using distributed AI.

**Human-computer interaction:**

Develop more natural and intuitive interfaces using parallel and distributed AI.

**Cognitive robotics:**

Control and coordinate robotic behavior using distributed AI.

**BENEFITS:**

- Improved scalability and flexibility
- Enhanced real-time processing capabilities
- More accurate and comprehensive cognitive models
- Better human-computer interaction and human-robot interaction
- Potential applications in fields like education, healthcare, and socialsciences.

**CHALLENGES:**

**Complexity:**

Managing and coordinating parallel and distributed processes

**Communication:**

Ensuring efficient data exchange between processing units.

**Synchronization:**

Coordinating tasks and maintaining consistency across processing units

**Scalability:**

Adapting to large-scale problems and datasets

**Interpretability:**

Understanding and explaining complex AI models and behaviors.

**PARALLELISM AND DISTRIBUTED IN REASONING SYSTEM**

Parallelism and distributed processing in reasoning systems refer to the use of multiple processing units or agents to perform reasoning tasks simultaneously, improving efficiency and scalability.

**PARALLELISM:**

1. **Task parallelism**: Dividing a reasoning task into smaller sub-tasks andexecuting them concurrently.

2. **Data parallelism**: Distributing data across multiple processing units andperforming the same reasoning task on each unit.

3. **Pipelined parallelism**: Breaking down a reasoning task into a series ofstages and executing them concurrently.

**DISTRIBUTED PROCESSING:**

1. **Decentralized reasoning**: Distributing reasoning tasks across multiple agents or nodes, each contributing to the overall solution.
2. **Distributed knowledge representation**: Storing knowledge acrossmultiple nodes, enabling efficient access and reasoning.
3. **Communication and coordination**: Ensuring data exchange and synchronization between nodes.

**BENEFITS:**

1. **Scalability**: Handle large-scale reasoning tasks and knowledge bases.
2. **Efficiency**: Reduce processing time through concurrent execution.
3. **Flexibility**: Adapt to changing environments and requirements.

**APPLICATIONS:**

1. **Artificial intelligence**: Enhance reasoning capabilities in AI systems.
2. **Expert systems**: Improve performance and scalability in expert systems.
3. **Multi-agent systems**: Enable distributed reasoning and decision-making.

**CHALLENGES:**

1. **Coordination and communication**: Manage data exchange and synchronization.
2. **Consistency and coherence**: Ensure consistent and coherent reasoningresults.
3. **Scalability and efficiency**: Balance computational resources

and reasoning performance.

**LEARNING CONNECTIONST MODEL:**

- The Connectionist Model in AI is a cognitive architecture that posits that cognitive processes can be understood in terms of the connections and interactions between simple computational units or nodes.
- This model is inspired by the structure and function of the brain and is often used to describe the processing of information in neural networks.

**KEY FEATURES:**

- **Distributed Representation**: Information is represented across multiple nodes, rather than being localized in a single location.
- **Parallel Processing**: Multiple nodes process information simultaneously, allowing for fast and efficient processing.
- **Learning and Adaptation**: Connections between nodes can be modified based on experience, allowing the system to learn and adapt.
- **Activation and Inhibition**: Nodes can be activated or inhibited by other nodes, allowing for complex patterns of activity to emerge.

**TYPES OF CONNECTIONST MODEL:**

- **Feed forward Networks**: Information flows only in one direction, frominput nodes to output nodes.
- **Recurrent Networks:** Information can flow in a loop, allowing forfeedback and recurrent processing.
- **Attractor Networks**: The network converges to a stable state, orattractor, which represents the processed information.

## APPLICATIONS OF CONNECTIONST MODEL:

- **Pattern Recognition**: Connectionist models can learn to recognizepatterns in data, such as images or speech.
- **Language Processing**: Connectionist models can learn to processand generate natural language.
- **Memory and Learning**: Connectionist models can learn and remember new information.
- **Decision-Making**: Connectionist models can make decisions basedon patterns and associations learned from data.

## BENEFITS OF CONNECTIONIST MODEL:

- **Flexibility**: Can be used to model a wide range of cognitive tasks and processes.
- **Scalability**: Can be applied to large-scale problems and datasets.
- **Biological Plausibility**: Is inspired by the structure and function of the brain.

## LIMITATIONS OF CONNECTIONIST MODEL:

- **Complexity:** Can be difficult to understand and analyze due to the complex interactions between nodes.
- **Lack of Interpretability**: Can be challenging to interpret the results of Connectionist models.
- **Training Requirements**: Requires large amounts of data and computational resources to train.

## HOPEFIELD NETWORKS

- Hopfield networks are a type of recurrent artificial neural network that serve as a content-addressable ("associative") memory system with binary threshold nodes.
- They are a simple example of a neural network that can store and recall memories.

## CHARACTERISTICS OF HOPEFIELD NETWORK:

- **Recurrent:** Hopfield networks have feedback connections, which allow the network to settle into a stable state.
- **Binary threshold nodes**: Each node in the network has a binary output (0 or 1) and a threshold value that determines its output.
- **Symmetric weights**: The weights connecting nodes are symmetric, meaning that the weight from node A to node B is the same as the weightfrom node B to node A.

## HOW HOPEFIELD NETWORKS WORK:

- **Initialization:** The network is initialized with a set of random weights and biases.
- **Training:** The network is trained on a set of patterns, where each pattern is a binary vector.
- **Storage:** The network stores the patterns in its weights and biases.
- **Recall:** When a noisy or incomplete pattern is presented to the network, it settles into a stable state that corresponds to the closest stored pattern.

## APPLICATIONS OF HOPEFIELD NETWORKS:

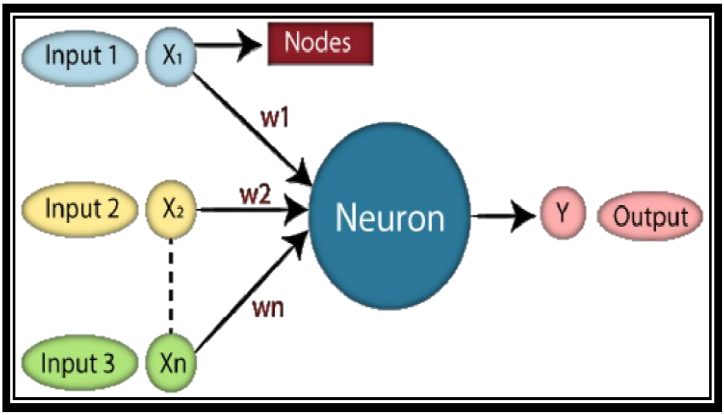
- **Associative memory:** Hopfield networks can store and recall patterns, making them useful for associative memory tasks.
- **Optimization:** Hopfield networks can be used to solve optimization problems by encoding the problem into a pattern and using the network to find the optimal solution.
- **Machine learning**: Hopfield networks can be used as a building block formore complex machine learning models.

**LIMITATIONS OF HOPEFIELD NETWORKS:**

- **Capacity:** Hopfield networks have a limited capacity for storing patterns.
- **Convergence:** The network may not always converge to a stable state.
- **Sensitivity to noise**: The network can be sensitive to noise in the input patterns.

**NEURAL NETWORKS:**

A Neural networks is a machine learning model inspired by the structure and function of the human brain. It consists of layers of interconnected nodes on "neurons". Which process and transmit information.




**Input layer:** Receives the data to be proceed.

**Hidden layer:** Performs complex calculation and transformation.

**Output layer:** Generates the final prediction on result.

**NEURAL NETWORK CAN:**

**Learn:** Adapt to new data and improve performance.

**Classify:** Categories data into classes or groups

**Predict:** Forecast future values or outcomes.

**Generate:** Create new data, like text or images.

**APPLICATIONS OF NEURAL NETWORKS:**

- Image recognition
- Natural language processing
- Speech recognition
- Predictive analysis.

**ADVANTAGES OF NEURAL NETWORKS:**

- **Adaptability:** Neural networks are useful for activities where the link between inputs and outputs is complex or not well defined because theycan adapt to new situations and learn from data.

- **Pattern Recognition:** Their proficiency in pattern recognition renders them efficacious in tasks like as audio and image identification, natural language processing, and other intricate data patterns.

- **Parallel Processing:** Because neural networks are capable of parallel processing by nature, they can process numerous jobs at once, which speeds up and improves the efficiency of computations.

- **Non-Linearity:** Neural networks are able to model and

comprehend complicated relationships in data by virtue of the non-linear activation functions found in neurons, which overcome the drawbacks of linear models.

**DISADVANTAGES OF NEURAL NETWORKS:**

- **Computational Intensity:** Large neural network training can be a laborious and computationally demanding process that demands a lot ofcomputing power.

- **Black box Nature:** As "black box" models, neural networks pose a problem in important applications since it is difficult to understand howthey make decisions.

- **Overfitting:** Overfitting is a phenomenon in which neural networks commit training material to memory rather than identifying patterns in data. Although regularization approaches help to alleviate this, the problemstill exists.

**TYPES OF NEURAL NETWORKS:**

- Convolutional neural networks
- Feedforward neural networks
- Recurrent neural networks
- Generative adversarial networks
- Modular neural networks

**Feed forward Neural Network**:

- The feedforward neural network is one of the most basic artificial neural networks.
- In this ANN, the data or the input provided travels in a single direction.
- It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist.
- So the feed forward neural network has a front-propagated wave

only and usually does not have back propagation.

**Convolutional Neural Network :**

- A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit.
- But a CNN has one or more than one convolutional layer that uses a convolution operation on the input and then passes the result obtained in the form of output to the next layer.
- CNN has applications in speech and image processing which is particularly useful in computer vision.

**Modular Neural Network:**

- A Modular Neural Network contains a collection of different neural networks that work independently towards obtaining the output with no interaction between them.
- Each of the different neural networks performs a different sub-task by obtaining unique inputs compared to other networks.
- The advantage of this modular neural network is that it breaks down a large and complex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.

**Radial basis function Neural Network:**

- Radial basis functions are those functions that consider the distance of a point concerning the center.
- RBF functions have two layers. In the first layer, the input is mapped into allthe Radial basis functions in the hidden layer and then the output layer computes the output in the next step.
- Radial basis function nets are normally used to model the data that represents any underlying trend or function.

**Recurrent Neural Network:**

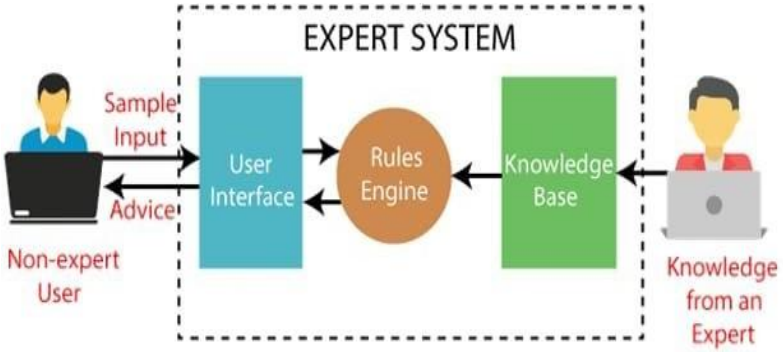
- The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer.
- The first layer in the RNN is quite similar to the feed-forward neural network and the recurrent neural network starts once the output of the first layer is computed.
- After this layer, each unit will remember some information from the previous step so that it can act as a memory cell in performing computations.

## UNIT 5 (EXPERT SYSTEM)

### What is Expert systems?

Expert systems are a crucial subset of artificial intelligence (AI) that simulate thedecision-making ability of a human expert. These systems use a knowledge base filled with domain-specific information and rules to interpret and solve complex problems. Expert systems are widely used in fields such as medical diagnosis, accounting, coding, and even in games.

### Block diagram of Expert System




### Examples of expert systems:

- CaDet (Cancer Decision Support Tool) is used to identify cancer in itsearliest stages.
- DENDRAL helps chemists identify unknown organic molecules.
- Dxplain is a clinical support system that diagnoses various diseases
- MYCIN identifies bacteria such as bacteremia and meningitis, andrecommends antibiotics and dosages.

- PXDES determines the type and severity of lung cancer a person has
- RI/XCON is an early manufacturing expert system that automaticallyselects and orders computer components based on customer specifications

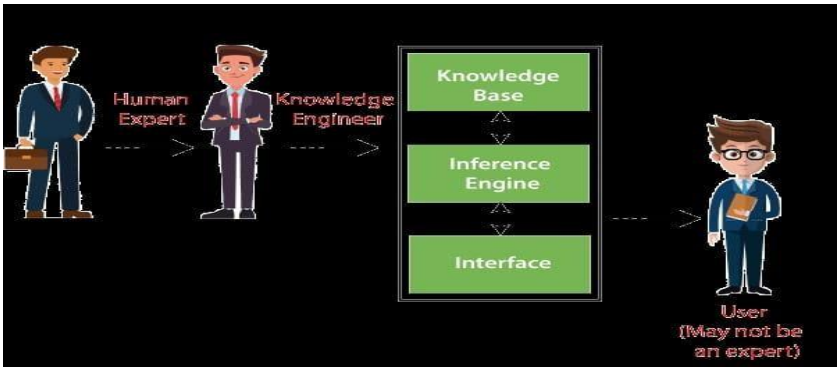
**Characteristics of Expert System:**

- 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 outputs 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 avery short period of time

**Componenets of Expert System:**

**An Expert System mainly consists of three Components**

- User Interface
- Inference Engine
- Knowledge Base




**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 inferenceengine.
- 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

**Inference 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.
- With the help of an Inference engine, the system extracts the knowledge from the Knowledge base.
- Two types of inference Engine

**1. Deterministic inference engine:**

The conclusions drawn from this type of inference engine are assumed to be true.It is based on facts and rules

**2. Probabilistic Inference engine:**

This type of inference engine contains uncertainity in conclusions, and based onthe probability.

**Knowledge Base**

- The Knowledge base 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.
- It is similar to a database that contains information and rules of a particulardomain 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.

**Capabilities of the Expert System**

- **Advising**: It is capable of advising the human being for the query of any domain from the particular ES
- **Provide decision-making Capabilities**: It provides the capability of decision making in any domain, such as for making any financial decision, decisions in medical science, etc.,
- **Demonstrate a device**: It is capable of demonstrating any new Products such as its features, specifications, how to use that product, etc.,
- **Problem-solving**: It has problem-solving, capabilities
- **Explaining a problem**: It is also capable of providing a detaileddescription of an input problem
- **Interpreting the input**: It is capable of interpreting the input given by theuser.

**Advantages of Expert System**

- These systems are highly reproducible.
- They can be used for risky places where the human presence is not safe.
- Error possibilities are less if the KB contains correct Knowledge.
- The performance of these systems remains steady as it is not affected byemotions, tension, or fatigue.
- They provide a very high speed to respond to a particular query.

**Limitations of Expert System**

- The response of the ES may get wrong if the knowledge base contains thewrong Information.
- Like a human being, it cannot produce a creative output for differentscenarios.
- Its maintenance and development cost, are very high.
- Knowledge acquisition for designing is much difficult.
- for each domain, we require a specific ES, which is one of the biglimitations.
- It cannot learn from itself and hence requires manual updates.

**Applications of Expert System**

- **Designing and manufacturing domain**: It can be broadly used for designing and manufacturing physical devices such as camera Lenses andautomobiles.
- **Knowledge Domain**: These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax

advisor.

- **Finance Domain**: In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not
- **Planning and scheduling**: The expert systems can also be used for Planning and scheduling some particular tasks for achieving the goal of that task.

**Types of Expert Systems:**

- Rule-Based Expert Systems.
- Frame-Based Expert Systems.
- Fuzzy Logic Systems.
- Neural Network -Based Expert Systems.
- Neuro-Fuzzy Expert Systems.

**1. Rule-Based Expert Systems:**

Use a set of "if-then" rules to process data and make decisions. These rules are typically written by human experts and capture domain-specific knowledge.

**Example**: MYCIN, an early system for diagnosing bacterial infections.

**2. Frame-Based Expert Systems:**

Represent knowledge using frames, which are data structures similar to objects in Programming. Each frame contains attributes and values related to a particularconcept.

**Example**: Systems used for knowledge Representation in areas Like NLP

**3. Fuzzy Logic Systems:**

It Handle uncertain or imprecise information using fuzzy logic, which allows forPartial truths rather than binary true false Values.

**Example**: Fuzzy control systems for managing household appliances Likewashing machines and air conditioners.

**4. Neural Network-Based Expert Systems**:

It use Artificial neural networks to learn from data and make predictions or decisions based on learned patterns. They are often used for tasks involving pattern recognition and classification.

**Example**: Deep Learning models for image and speech recognition.

**5. Neuro-Fuzzy Expert Systems:**

Neural networks and fuzzy logic to combine the learning capabilities of neural networks with the handling of uncertainity and imprecision offered by fuzzy logic.

This hybrid approach helps in dealing with complex problems where both patternrecognition and uncertain reasoning are required.

**Example:** Automated control systems that adjust based on uncertain environmental conditions or financial forecasting models that handle both quantitative data and fuzzy inputs.

**How Expert Systems Work?**

- Input Data: Users provide data or queries related to a specific problem orscenario.
- Processing: The inference engine processes the input data using the rules in the knowledge base to generate conclusions or recommendations.

- Output: The system presents the results or solutions to the user through theuser interface.

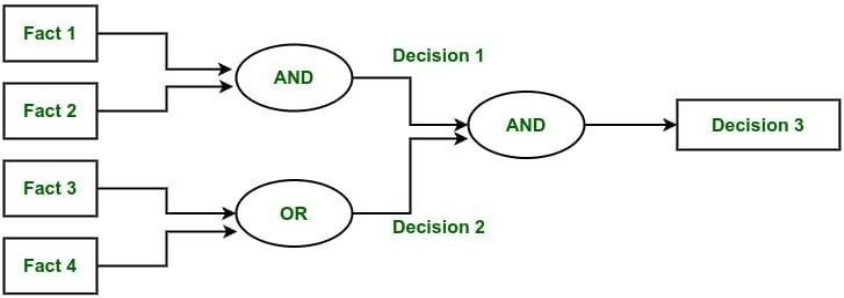
**Explanation:** If applicable, the system explains how the conclusions werereached, providing Insights into the reasoning process.

**Fundamental Methods of Expert System**

- Forward chaining
- Backward chaining

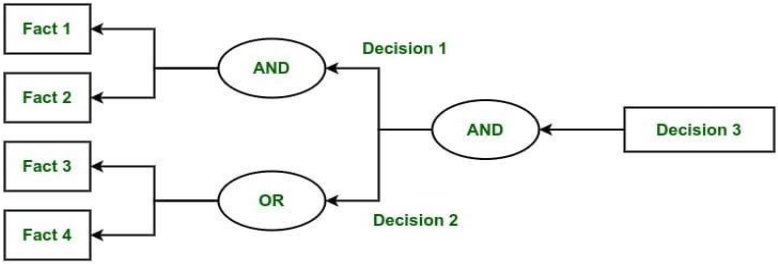
**1. Forward chaining:**

- "What can happen next?" The inference engine follows the chain of conditions and derivations and finally deduces the outcome.
- It considers all the facts and rules, and sorts them before concluding to asolution
- This strategy is followed for working on conclusion, result, or effect.
- Example, prediction of share market status as an effect of changes ininterest rates.



**2. Backward chaining:**

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




**COMMON SENSE**

- Common sense is the mental skills that most people share.
- Common Sense is ability to analyze a situation based on its context, using millions of integrated pieces of common knowledge.
- Common sense is what people come to know in the process of growing andliving in the world.
- Common sense knowledge includes the basic facts about events and their effects, facts about knowledge and how it is obtained, facts about beliefs and desires. It includes the basic facts about material objects and their properties
- Example: Everyone knows that dropping a glass of water, the glass will break and water will spill on podium.

However, this information is not Obtained by formula or equation for a falling body or equations governingfluid flow.

- The goal of the formal common sense reasoning community is to encode this implicit knowledge using format logic Common sense is Identified as:
    - Common sense knowledge: What every one know
    - Common sense reasoning: ability to use common sense knowledge

**Common sense knowledge:**

What one can express as a fact using a richer ontologyExample:

1. Every person is younger than the Person's mother
2. If you hold a knife by it's blade then the blade may cut you.
3. If you drop paper into a flame then the paper will burn.

**Common sense Reasoning:**

What one builds as a reasoning method into his program.

**Example:**

- If you have a problem, think of a past situation where you solved a similarproblem
- If you fall at something, imagine how you might have done thingsdifferently.

**Common sense Architecture:**

- The system takes as input a template Produced by information extractionsystem about certain aspects of a scenario

- The template is a frame with slots and slots fillers
- The template is fed to a script classifier, which classifies what script is active in the template
- The template and the script are passed to a reasoning problem builder specific to the script, which converts the template into a commonsense reasoning problem.
- The problem and a commonsense knowledge base are passed to a commonsense reasoner. It infers and fills in missing details to Produce a model of the input text.
- The model provides a deeper representation of the input, than is provided by the template alone.

**Importance of Common Sense in Expert System**

- Improved decision-making
- Enhanced problem-solving
- Increased accuracy
- Better understanding of human behavior
- More natural human-computer interaction

**Challenges of Implementing Common Sense**

- Knowledge Acquisition
- Representation and organization
- Integration with domain - specific knowledge
- Reasoning and inference
- Evaluation and validation

**Techniques for Implementing Common Sense**

- Knowledge graph-based approaches
- Machine learning and deep Learning

- Natural Language Processing and understanding
- Cognitive architectures and modeling
- Hybrid approaches

**Applications of common Sense**

**Natural language Processing & generation**

- Computer vision and image understanding.
- Robotics and autonomous systems
- Expert systems and decision support
- Human-computer interaction & dialogue systems

**Qualitative physics**

- Qualitative physics in AI is a way to study the physical world byrepresenting and reasoning about it.
- Qualitative physics uses dimensional analysis to solve problems, which requires Knowledge of physical variables and dimensional representation.
- This allows reasoning about systems and devices without needing to knowthe physical laws that govern them.
- for eg:, if a problem involves time period, mass, and spring constant, thevariables and their dimensions are t (T), m [M], K[MT-2].
- Qualitative Physics can also be used to solve physical system problems, and can be used for tasks like behavior analysis and conceptual design.

**Techniques used in Qualitative physics**

- Qualitative simulation: Simulating physical phenomena using qualitative models, such as qualitative differential equations

- Model-based reasoning: Using qualitative models to reason about physicalsystems and predict behaviors.
- Case-based reasoning: Storing and retrieving cases Physical phenomena toreason about new situations
- Ontologies: Representing physical knowledge using ontologies, which provide a framework for organizing and reasoning about qualitative knowledge

**Applications of Qualitative Physics**

- Robotics: Enables robots to understand and interact with their physicalenvironment
- Computer vision: Allows for qualitative understanding of visual scenes andevents
- Natural Language Processing: Enables understanding of physical phenomena described in Natural language.
- Expert Systems: Provides a foundation for building expert systems thatreason about physical domains

**Benefits of Qualitative Physics**

- Improved explainability: Provides insights into Physical phenomena andsystem behaviors.
- Robustness. Less sensitive to numerical errors and uncertainties
- Flexibility: Can handle incomplete or uncertain knowledge
- Efficiency: Reduces computational complexity in certain applications

**COMMON SENSE ONTOLOGIST**

- An Ontologist is a professional who specializes in designing, developing, and managing ontologies.

- The primary goal is to create a structured and comprehensive representation of Knowledge that can be easily understood and utilized byboth humans and machines.
- Ontologies are used to facilitate Knowledge sharing, communication,collaboration between humans and machines.

**How ontologies used in AI?**

In AI, ontologies are used to model Knowledge about the world in a structured form. This allows AI systems to understand and reason about the relationships between concepts, improving their ability to process natural language, make decisions, and learn from data

**Applications of Ontologists in AI**

- Expert Systems: Ontologists in AI develop Ontologies that power expert systems, which mimic human decision-making in a particular domain
- Natural Language Processing: Ontologists in AI use ontologies to improve natural language processing systems, enabling machines to understand human language
- Decision Support Systems: It support decision-making in complex domains
- Data Integration: Ontologists in AI use ontologies to integrate data from multiple sources, enabling machines to reason about the integrated data.
- Robotics and autonomous system: It enable robots and autonomous systems to understand their environment and make decisions

**Key Components of Ontology**

- Concepts: Representing entities, objects, or ideas within the domain

- Relationship: Defining how concepts interact, relate, or connect
- Rules: Specifying constraints, Logic, or axioms governing the domain
- Axioms: Fundamental truths or assumptions underlying the ontology
- Instances: specific examples or individuals within the domain.

**Benefits of ontology**

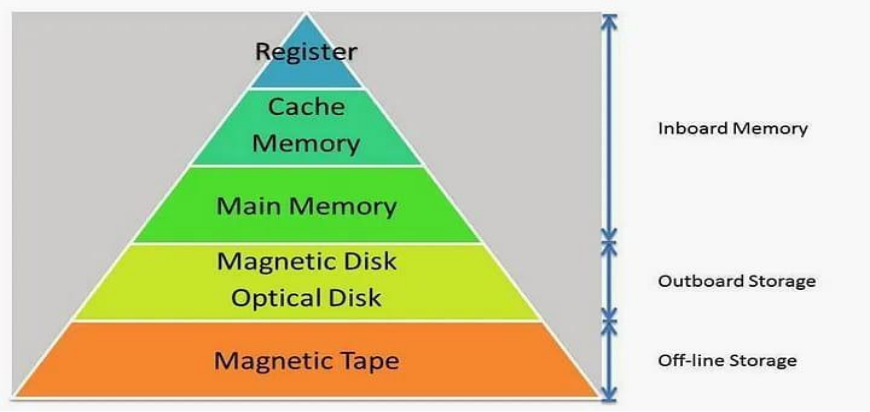
- Improved knowledge organization and sharing
- Enhanced reasoning & decision-making capabilities.
- Increased data consistency and quality
- Better interoperability and integration
- More explainable and transparent AI Systems

## MEMORY ORGANIZATION

Memory Organization provides a framework within which to understand the relationship between individual identification, duplicate records, and the proper functioning of AI

### Types of Memory Organization

There are several types of memory Organization used in computer system.




**Memory Organization in Artificial Intelligence can include a variety oftechniques that help AI Systems.**

1. E-MDPs: Episodic Memory Organization packets. A network of frames that contain conceptual information about different types of Episodicevents.

2. Replay memory: A technique used in AI, Particularly in reinforcement learning, where an agent learns to make decisions by interacting with its environment.

3. Working memory: Another critical aspect of AI inspired by the human cognitive System. Memory is central to common sense behavior and also the basis for learning. Human memory is still not fully understood however Psycologists have proposed several ideas

4. Short-Term Memory (STM):

- Only a few items at a time can be held here.
- Perceptual information stored directly here

5. Long-Term Memory (ITM):
    - Capacity for storage is very large and fairly permanent
    - LTM is often divided up further

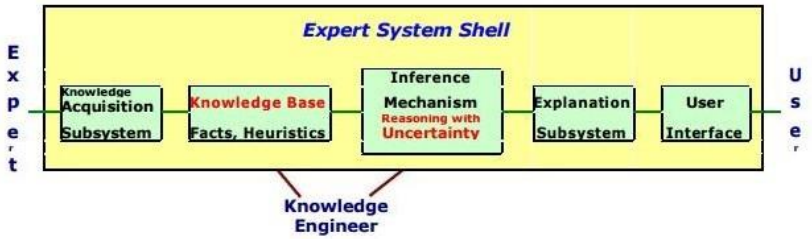
Episodic Memory: Contains information about Personal experiences
Semantic Memory: General facts with no Personal meaning

Three distinct Memory Organizations Packets (MOPS) code Knowledgeabout an sequence

- 1st MOP represents the physical sequence of events
- 2nd MOP represents the set of social events that takes place
- 3rd MDP revolves around the goals of the Person in the particular episode

**EXPERT SYSTEM SHELLS**

- Expert System shells are a collection of software packages and tools usedto develop expert systems
- A shell provides the developers with Knowledge acquisition, inferenceengine, user interface, explanation facility.
- Initially each expert system is build from Scratch (LISP)
- Example of shell is EMYCIN (for Empty- MYCIN derived from MYCIN)
- Expert System shells CUPS, JESS, DROOLS




**Expert System has two main parts**

- A Knowledge Base: This is where all the expert info is stored
- An inference Engine: This part uses the stored knowledge to figure outsolutions

**Benefits of Expert System Shell**

Time and Cost Savings: Less time spent on development means Loweroverall costs

- **User - Friendly Interface:**

    1. Easier Knowledge: input we can add expert knowledge without complexcoding

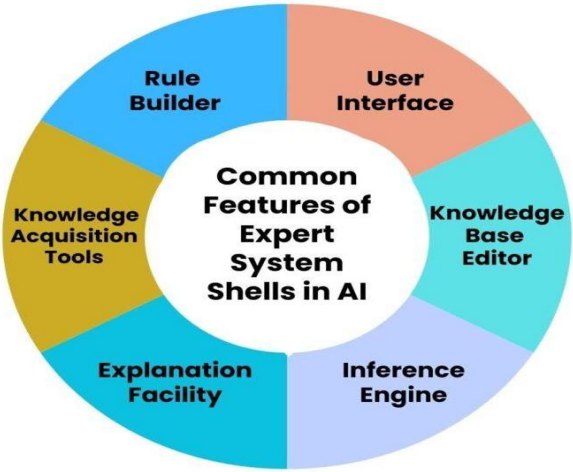
    2. Simple maintenance: Updating the system is straight forward, even fornon-technical users

- **flexibility and Customization:**

    1. Versatile applications: We can use them for different industries likehealthcare, finance or engineering
    2. Easy modifications: As your needs change, you can quickly adjust yourexpert system

**Build in Features**

- Inference Engines: It They are already included, so you don't need to buildOne from scratch.
- Explanation facilities: These help users understand how the system reachesits conclusions
- Knowledge base Editors: They make organizing and updating informationsimple




**User Interface:**

- Easy-to-use screens for inputting data and getting results
- Often includes menus, buttons, forms
- Helps non-technical users interact with the system

**Knowledge Base Editor**:

- A tool for adding and organizing expert Knowledge
- Allows, users to input rules, facts and relationships
- Makes it easy to update information as needed

**Explanation Facility:**

- Helps users understand how the system reached its decision
- shows the steps and rules used in the reasoning process
- Builds trust by making the system's logic transparent

**Inference Engine:**

- The "brain" of the Expert System
- Uses the knowledge base to solve problems or make decision
- Applies logical reasoning to reach conclusions

**Knowledge Acquisition Tools:**

- Assist in gathering and structuring expert Knowledge
- May include interview templates or questionnaries
- Helps convert human expertise into a format the system can use

**Rule Builder:**

- Allows users to create if then rules easily
- Helps in setting up the logic the system will follow
- often includes a visual interface for creating rules.

Examples of Expert System Shells

**CLIPS (C Language Integrated Production System)**

- Developed by NASA
- Free and open-source
- Good for building rule-based expert systems
- Used in space shuttle operations

**Jess (Java Expert System Shell):**

- Based on CLIPS but works with Java
- Fast and powerful

- popular in academic and research settings

**Prolog:**

- It is not just a shell but is often used to build expert systems
- Good for logical reasoning tasks
- Used in NLP

**Drools:**

- Open-source business rules management system.
- Integrates well with Java applications
- Used in many industries for decision automation

**Future of Expert System Shells in AI**

- The future of ES shells in AI looks bright and full of possibilities. We can expect to see these tools become more powerful and user-friendly.
- They will likely merge with machine Learning, creating systems that notonly use expert knowledge, but also team and adapt
- As AI continues to evolve, these shells could play a crucial role in developing. more advanced and flexible systems across Various fields

**REFERENCES:**

1. Elaine Rich, Kevin Knight, “Artificial Intelligence”, 3/e, Tata McGraw Hill, 2017.
2. Russell, “Artificial intelligence: A modern Approach, Pearson Education ,3rd edition,2013
3. Gupta, G. Nagpal, "Artificial Intelligence and Expert Systems", Mercury Learning & Information, 2020.
4. C.S. Krishnamoorthy, S. Rajeev, "Artificial Intelligence and Expert Systems for Engineers", CRC Press, 2018.
5. V. Daniel Hunt, "Artificial Intelligence & Expert Systems Sourcebook, Springer US, 2012.
6. Artificial Intelligence and Expert system by V.Daniel hunt, Springer press, 2011.
7. Nilsson N.J., “Principles of Artificial Intelligence”, Morgan Kaufmann.1998.