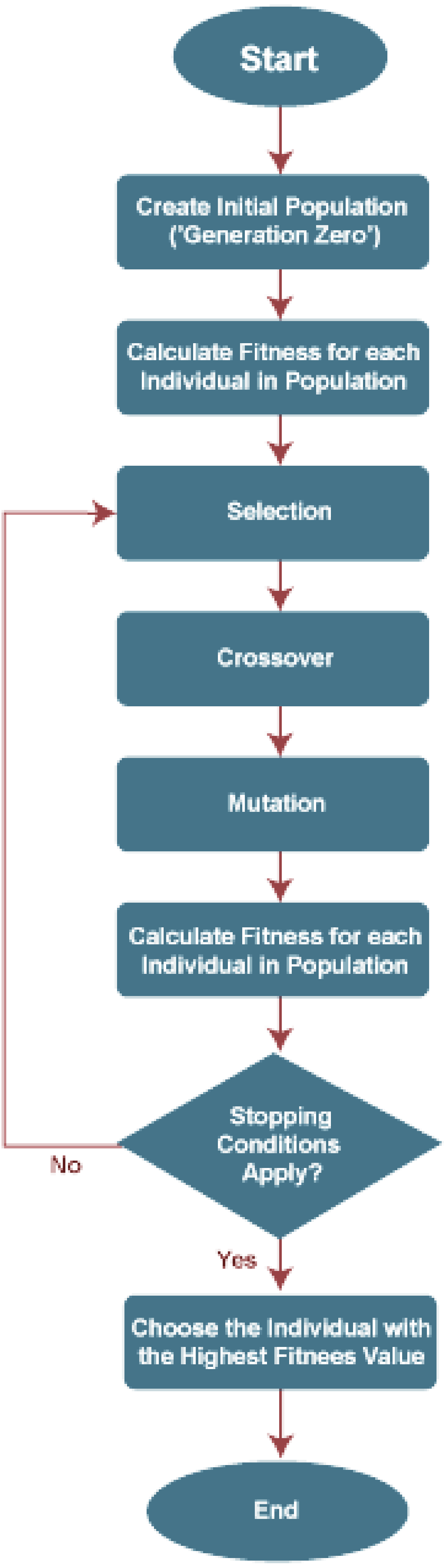
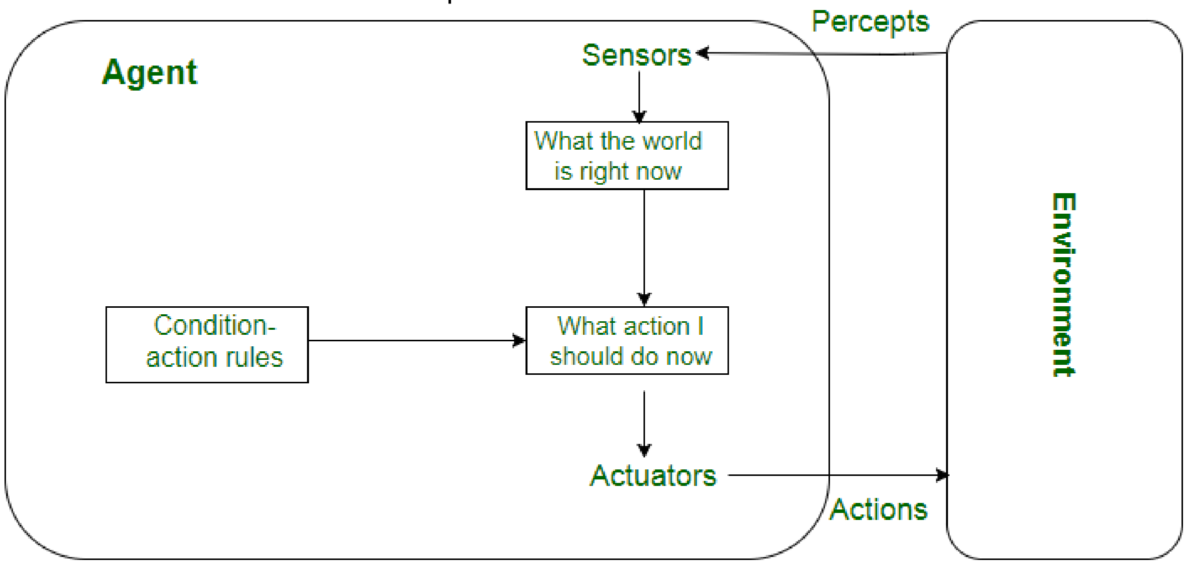
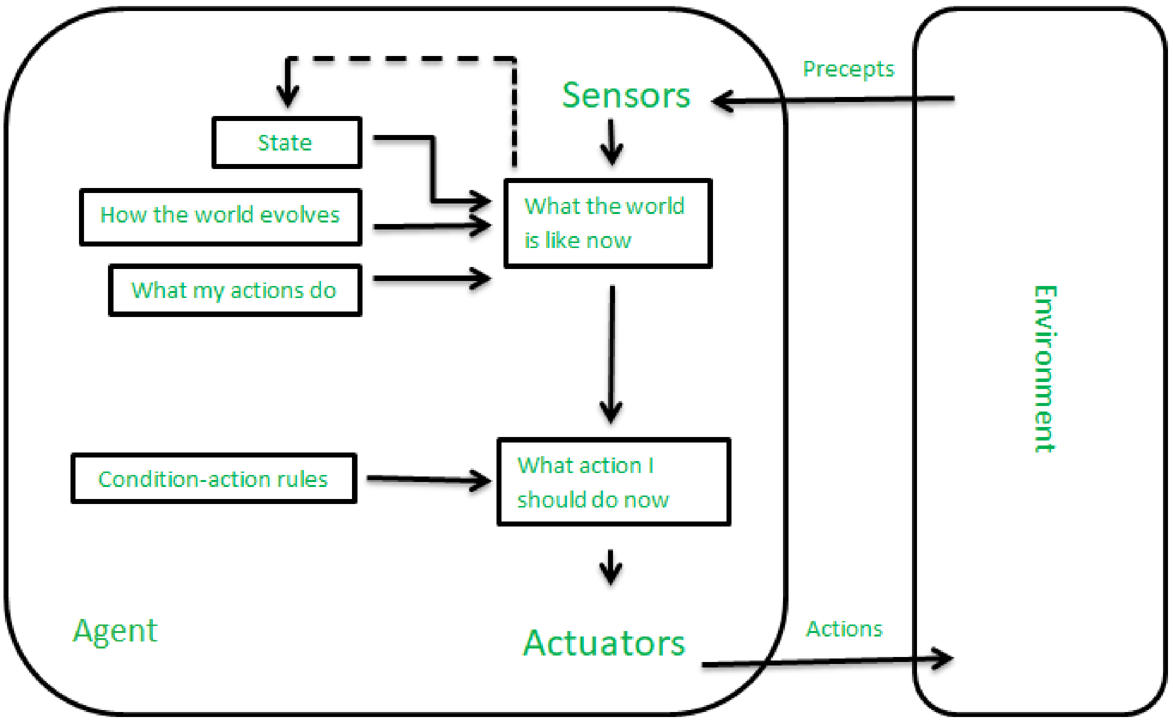
Genetic Algorithm: It is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature." It is used to solve optimization problems in machine learning.
Population: Population is the subset of all possible or probable solutions, which can solve the given problem.
Chromosomes: A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.
Gene: A chromosome is divided into a different gene, or it is an element of the chromosome.
Allele: Allele is the value provided to the gene within a particular chromosome.
Fitness Function: The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.
Genetic Operators: In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.
Selection: After calculating the fitness of every existent in the population, a selection process is used to determine which of the individualities in the population will get to reproduce and produce the seed that will form the coming generation.
Working: 1) Initialization: The process of a genetic algorithm starts by generating the set of individuals, which is called population. Here each individual is the solution for the given problem. An individual contains or is characterized by a set of parameters called Genes. Genes are combined into a string and generate chromosomes, which is the solution to the problem. One of the most popular techniques for initialization is the use of random binary strings. **2) Fitness Assignment:** Fitness function is used to determine how fit an individual is? It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function. The fitness function provides a fitness score to each individual. This score further determines the probability of being selected for reproduction. The high the fitness score, the more chances of getting selected for reproduction. **3) Selection:**The selection phase involves the selection of individuals for the reproduction of offspring. All the selected individuals are then arranged in a pair of two to increase reproduction. Then these individuals transfer their genes to the next generation.
Types 1. Roulette wheel selection **2.**Event selection **3.**Rank- grounded selection
4) Reproduction: After the selection process, the creation of a child occurs in the reproduction step. In this step, the genetic algorithm uses two variation operators that are applied to the parent population. The two operators involved in the reproduction phase:
A)Crossover: The crossover plays a most significant role in the reproduction phase of the genetic algorithm. In this process, a crossover point is selected at random within the genes. Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing the offspring. **Eg: P1:** A**BCDE**FGH with P2: **FGH**ADBE**A**→ **Offspring: FGH**BCDE**A** The genes of parents are exchanged among themselves until the crossover point is met. These newly generated offspring are added to the population. This process is also called or crossover. **Types of crossover styles: 1.**One point crossover **2.** Two-point crossover **3.** Livery crossover **4.**Inheritable Algorithms crossover. **B) Mutation:** The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population. It can be done by flipping some bits in the chromosomes. Mutation helps in solving the issue of premature convergence and enhances diversification. The below image shows the mutation process: **Types of mutation: 1.**Flip bit mutation **2.**Gaussian mutation **3.**Exchange/Swap mutation. **Eg: Before:** FG**H**BCDE**A**;
After: FG**M**BCDE**N** **5) Termination:** After the reproduction phase, a stopping criterion is applied as a base for termination. The algorithm terminates after the threshold fitness solution is reached. It will identify the final solution as the best solution in the population.
Advantages:1) The parallel capabilities of genetic algorithms are best. 2) It helps in optimizing various problems such as discrete functions, multi-objective problems, and continuous functions. 3) It provides a solution for a problem that improves over time. 4)Genetic algorithm does not need derivative information. **Limitations: 1)** Genetic algorithms are not efficient algorithms for solving simple problems. 2) It does not guarantee the quality of the final solution to a problem. 3) Repetitive calculation of fitness values may generate some computational challenges.


Start
Create Initial Population ('Generation Zero')
Calculate Fitness for each Individual in Population
Selection
Crossover
Mutation
Calculate Fitness for each Individual in Population
Stopping Conditions Apply?
No
Yes
Choose the Individual with the Highest Fitnees Value
End

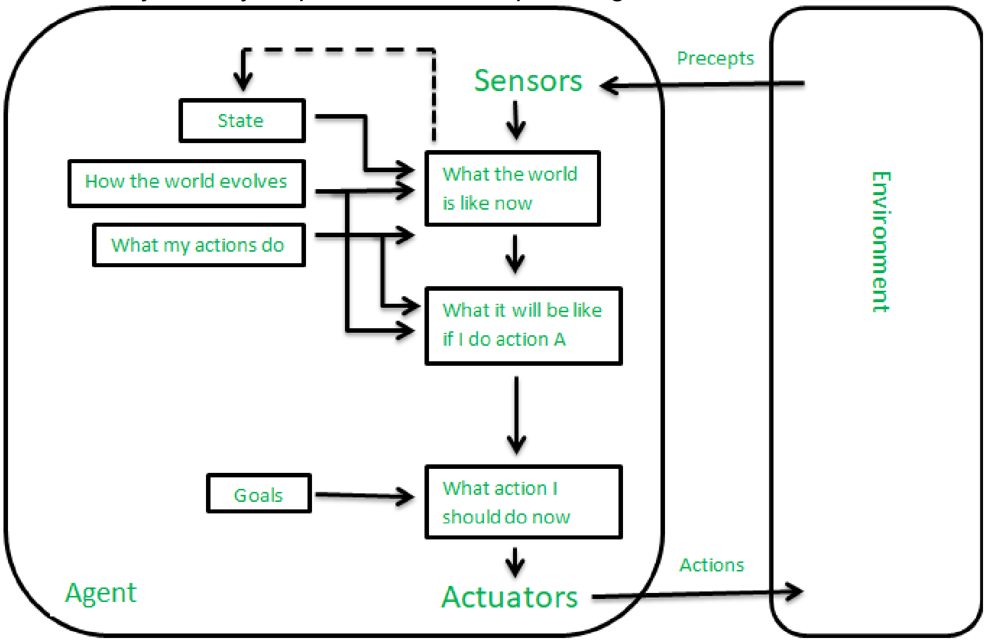
Simple reflex agents: Simple reflex agents ignore the rest of the percept history and act only on the basis of the current percept which is the history of all that an agent has perceived to date. The agent function is based on the condition-action rule that maps a state i.e, condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable.
Limitations: Very limited intelligence. No knowledge of non-perceptual parts of the state. Usually too big to generate and store. If there occurs any change in the environment, then the collection of rules need to be updated.



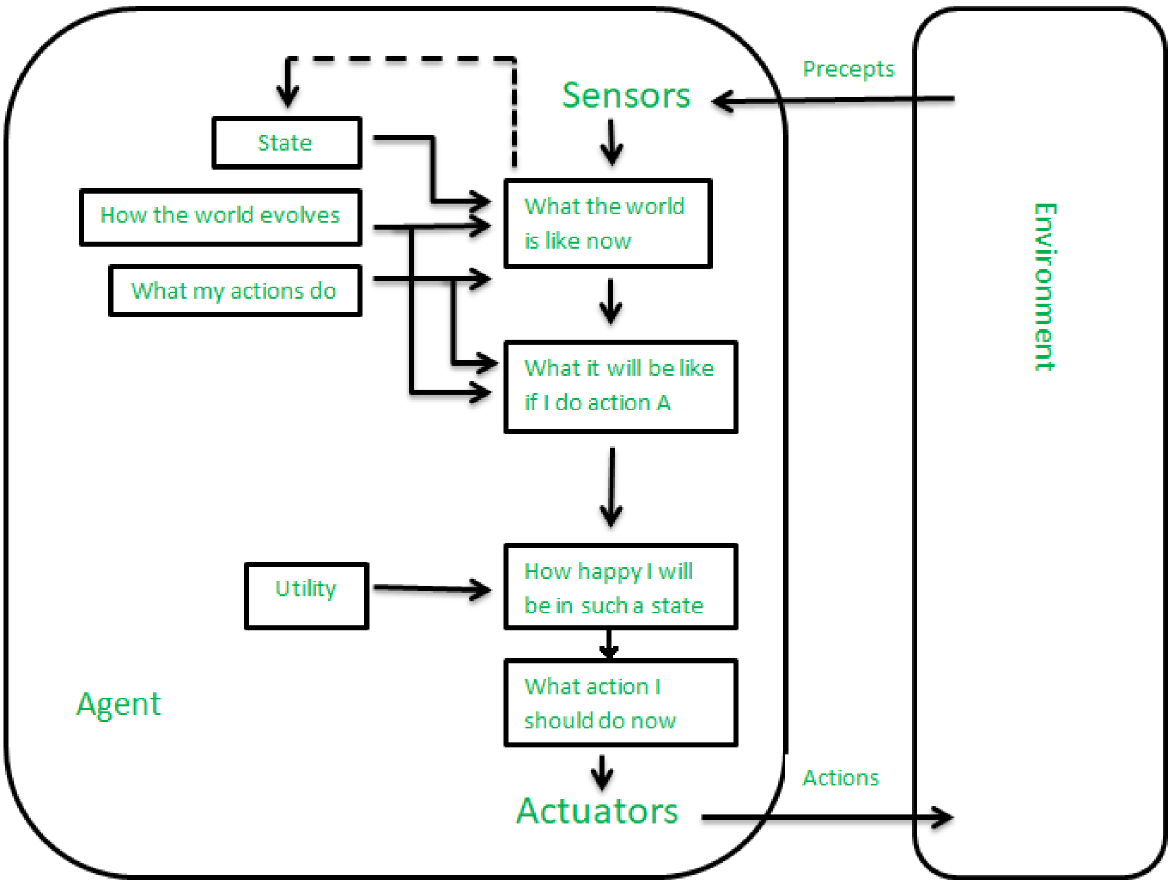
Model-based reflex agents: It works by finding a rule whose condition matches the current situation. A model-based agent can handle partially observable environments by the use of a model about the world. The agent has to keep track of the internal state which is adjusted by each percept and that depends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



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

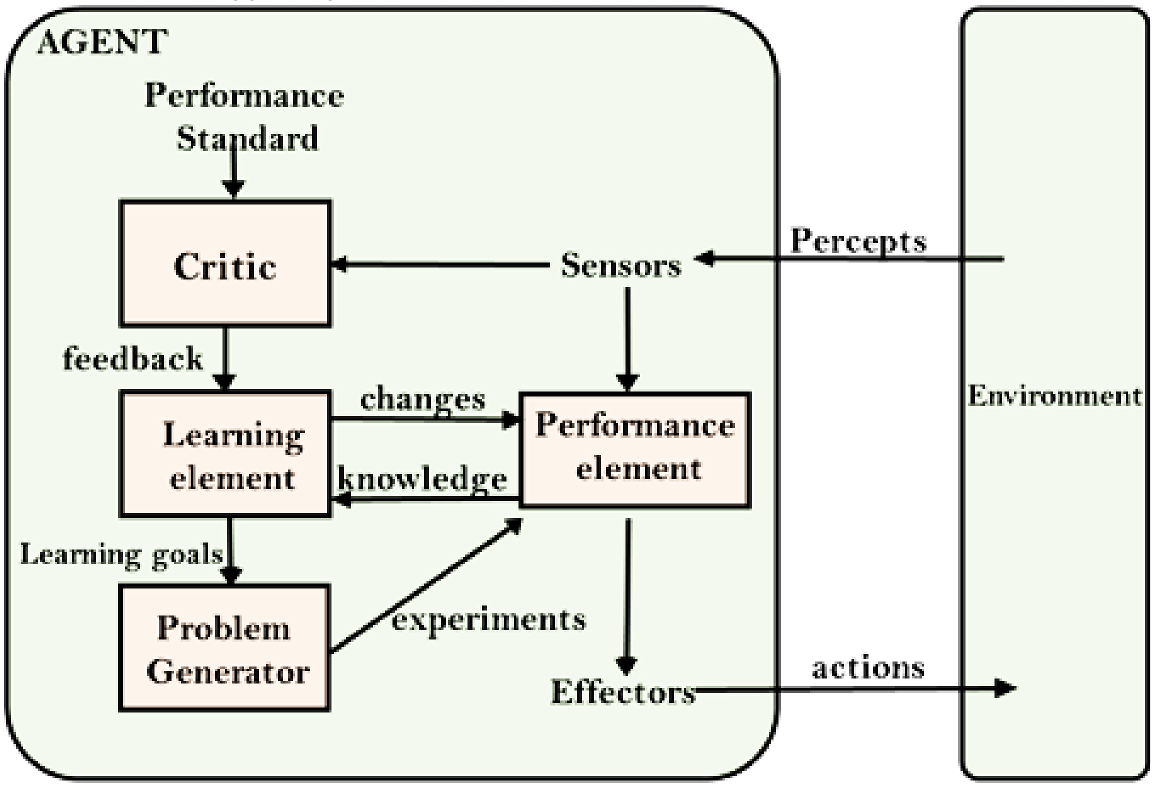


Utility-based agents: The agents which are developed having their end uses as building blocks are called utility-based agents. When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a preference (utility) for each state.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



State
How the world evolves
What my actions do
Sensors
Precepts
What the world is like now
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Actions
Environment

Learning Agent : A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. four conceptual components: **Learning element:** It is responsible for making improvements by learning from the environment. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard. **Performance element:** It is responsible for selecting external action. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



Agent is a computer program or system that is designed to perceive its environment, make decisions and take actions to achieve a specific goal or set of goals. The agent operates autonomously, meaning it is not directly controlled by a human operator. An agent is anything that can perceive its environment through sensors and acts upon that environment through effectors.
Uses of Agents :1) Robotics: Agents can be used to control robots and automate tasks in manufacturing, transportation, and other industries. **2) Smart homes and buildings:** Agents can be used to control heating, lighting, and other systems in smart homes and buildings, optimizing energy use and improving comfort. **3) Transportation systems:** Agents can be used to manage traffic flow, optimize routes for autonomous vehicles, and improve logistics and supply chain management. **4) Healthcare:** Agents can be used to monitor patients, provide personalized treatment plans, and optimize healthcare resource allocation.
5) Finance: Agents can be used for automated trading, fraud detection, and risk management in the financial industry. **6) Games:** Agents can be used to create intelligent opponents in games and simulations, providing a more challenging and realistic experience for players. **7) Natural language processing:** Agents can be used for language translation, question answering, and chatbots that can communicate with users in natural language.
8) Cybersecurity: Agents can be used for intrusion detection, malware analysis, and network security. **9) Environmental monitoring:** Agents can be used to monitor and manage natural resources, track climate change, and improve environmental sustainability.
10) Social media: Agents can be used to analyze social media data, identify trends and patterns, and provide personalized recommendations to users.

Diff b/w model based agent and Utility based agent: **1) Rely on internal representations of the environment and use models to make decisions.** Focus on maximizing a utility function or achieving a specific goal. **2) Have an explicit model of the environment, including knowledge about its dynamics and possible actions.** Do not require an explicit model of the environment; they rely on heuristics or utility functions. **3) Use the model to simulate the consequences of different actions and select the one that leads to the desired outcome.** Evaluate actions based on their utility or desirability and choose the one with the highest utility. **4) Can make decisions based on predictions of the future state of the environment.** Evaluate the current state of the environment and select actions that maximize the expected utility at that moment. **5) Suitable for domains where the environment is known and predictable, and accurate modeling is possible.** Suitable for domains where the environment is uncertain, dynamic, or complex, and modeling is challenging. **6) Eg: planning systems, chess-playing programs & autonomous vehicles.** Eg: reinforcement learning agents, economic agents & decision-making systems.

Diff b/w model based agent and Goal based agent: **1) Rely on internal representations of the environment and use models to make decisions.** Focus on achieving predefined goals without relying on explicit models of the environment. **2) Have an explicit model of the environment, including knowledge about its dynamics and possible actions.** Do not require an explicit model of the environment; they focus on the desired outcome rather than modeling the environment. **3) Use the model to simulate the consequences of different actions and select the one that leads to the desired outcome.** Define goals and work towards achieving them, often by applying predefined rules or heuristics. **4) Can make decisions based on predictions of the future state of the environment.** Evaluate the current state and compare it against the desired goals to determine the next action. **5) Suitable for domains where the environment is known and predictable, and accurate modeling is possible.** Suitable for domains where the environment is less predictable or complex, and the focus is on goal attainment. **6) Eg: planning systems, chess-playing programs, and autonomous vehicles.** Eg: rule-based systems, expert systems, and problem-solving agents.

Diff b/w Utility based agent and Goal based agent: **1) Focus on maximizing overall utility or value in decision-making.** Focus on achieving predefined goals or objectives. **2) Assign a numerical value or utility to each possible action or outcome.** Evaluate actions based on their contribution to the goal attainment. **3) Consider trade-offs and make decisions based on the expected utility of each action.** Evaluate actions based on their desirability or relevance to the predefined goals. **4) Can handle situations where multiple goals or objectives exist, and trade-offs need to be made.** Suitable for domains with well-defined goals and where the focus is on achieving those specific goals. **5) Take into account uncertainties and probabilities when computing expected utility.** May not explicitly consider uncertainties or probabilities, but rather prioritize actions based on their alignment with the goals. **6) Eg: decision-making systems, economic models, and resource allocation systems.** Eg: rule-based systems, expert systems, and problem-solving agents.

Diff b/w problem solving and planning agent: **1) Focus on finding solutions to specific problems or tasks.** Focus on generating a sequence of actions to achieve a desired goal or objective. **2) Analyze the current state, goal state, and possible actions to determine the best course of action.** Construct a plan by considering the initial state, goal state, and a set of possible actions. **3) Typically operate in dynamic environments with changing states and goals.** Typically operate in static environments with fixed states and goals. **4) Employ various problem-solving techniques, such as search algorithms, heuristics, and problem decomposition.** Employ planning algorithms, such as state-space search, STRIPS, or hierarchical task networks (HTN). **5) React to changes in the environment and adapt their problem-solving approach as needed.** Pre-determine a sequence of actions in advance to achieve a specific goal. **6) Eg: expert systems, intelligent tutoring systems, and diagnostic systems.** Eg: automated planning systems, robotics, and autonomous vehicles.

Diff b/w Informed(Heuristic Search) and uninformed search (Blind Search) algorithms:
1) Utilize additional information about the problem domain to guide the search process. Do not have any additional information about the problem domain and make decisions solely based on the available information. **2) Make use of heuristic functions or domain-specific knowledge to estimate the desirability of states or actions.** Do not employ heuristic functions and treat all states equally during the search process.
3) Generally more efficient as they can prioritize the search towards more promising paths based on the heuristic information. May explore a larger portion of the search space since they lack the guidance provided by heuristic functions. **4) It helps find the solution quickly.** It takes more time to show the solution. **5) It may or may not be complete.** It is always complete. **6) It is inexpensive.** It is expensive. **7) It consumes less time.** It consumes moderate time. **8) It gives the direction about the solution.** There is no suggestion regarding finding the solution. **9) It is less lengthy to implement.** It is lengthy to implement.1**0) Examples include A* search, Best-First search, and Greedy search.** Examples include Breadth-First search, Depth-First search, and Uniform Cost search.

PEAS of **Wumpus world: Performance measure:** +1000 reward points if the agent comes out of the cave with the gold. -1000 points penalty for being eaten by the Wumpus or falling into the pit. -1 for each action, and -10 for using an arrow. The game ends if either agent dies or came out of the cave.
Environment: A 4*4 grid of rooms. The agent initially in room square [1, 1], facing toward the right. Location of Wumpus and gold are chosen randomly except the first square [1,1]. Each square of the cave can be a pit with probability 0.2 except the first square.
Actuators: Left turn; Right turn; Move forward; Grab; Release; Shoot.
Sensors: The agent will perceive the stench if he is in the room adjacent to the Wumpus. (Not diagonally). The agent will perceive breeze if he is in the room directly adjacent to the Pit. The agent will perceive the glitter in the room where the gold is present. The agent will perceive the bump if he walks into a wall. When the Wumpus is shot, it emits a horrible scream which can be perceived anywhere in the cave. These percepts can be represented as five element list, in which we will have different indicators for each sensor. Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as: **[Stench, Breeze, None, None, None].**
The Wumpus world Properties: 1) Partially observable: The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room**. 2)Deterministic: A**s the result and outcome of the world are already known.
3) Sequential: The order is important, so it is sequential. **4) Static:** It is static as Wumpus and Pits are not moving. **5) Discrete:** The environment is discrete. **6) One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

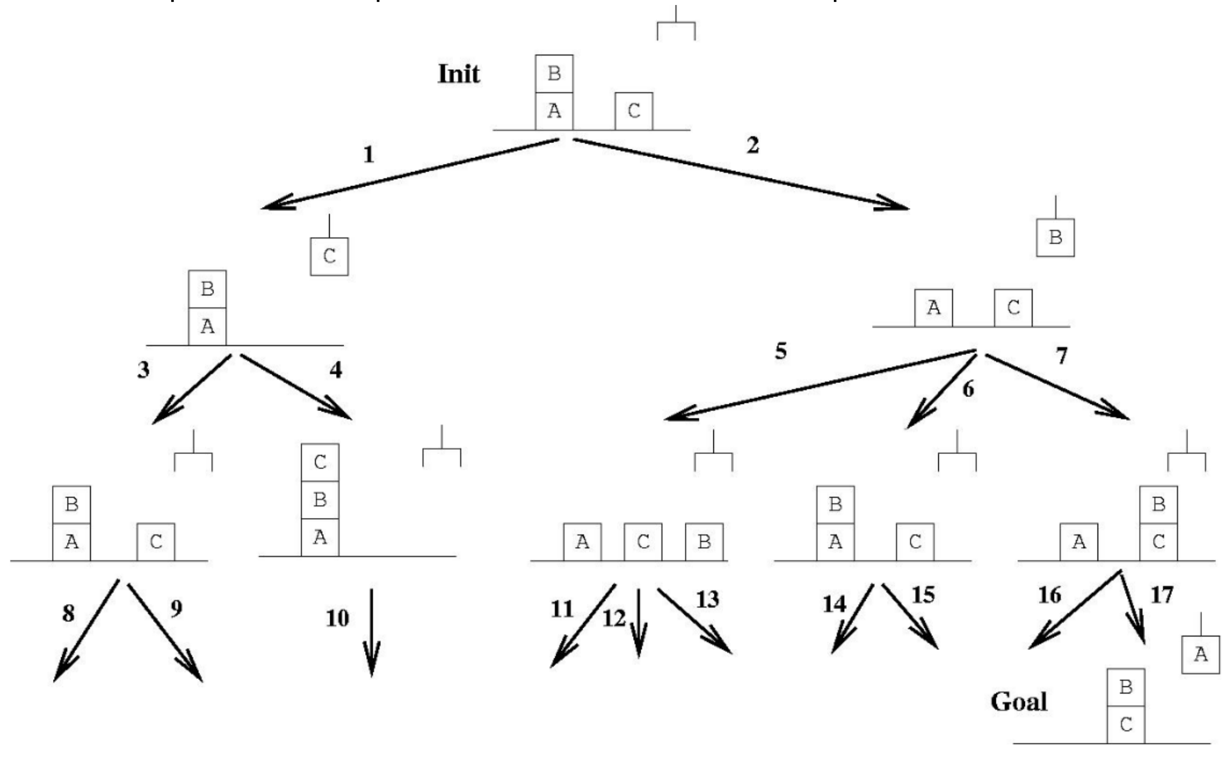
Intelligent Agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example. **Draw diag of simple reflex agent.**
Rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions. A rational agent is said to perform the right things. For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward. **Rationality of an agent** is measured by its performance measure. **Judged upon: 1.** Performance measure which defines the success criterion. **2.**Agent prior knowledge of its environment. **3.**Best possible actions that an agent can perform. **4.**Sequence of percepts.
Structure of IA: Agent = Architecture + Agent program; **Architecture:** is machinery that an AI agent executes on. **Agent Function:** is used to map a percept to an action. **Agent program:** is an implementation of agent function. An agent program executes on the physical architecture to produce function f. **f:P* → A**

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model.
P: Performance measure; E: Environment; A: Actuators; S: Sensors
Let's suppose a **self-driving car** then PEAS representation will be:
Performance: Safety, time, legal drive, comfort
Environment: Roads, other vehicles, road signs, pedestrian
Actuators: Steering, accelerator, brake, signal, horn
Sensors: Camera, GPS, speedometer, odometer, accelerometer, sonar.

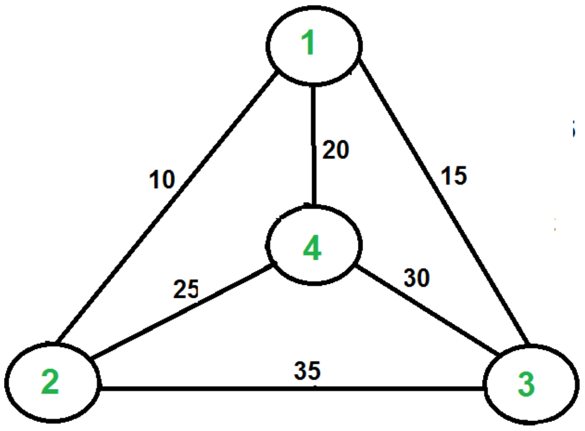
Properties of agent task environment 1. Fully observable vs Partially Observable: If an agent sensor can sense or access the complete state of an environment at each point of time then it is a fully observable environment, else it is partially observable. A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world. An agent with no sensors in all environments then such an environment is called as unobservable. **Eg:** Chess – the board is fully observable, and so are the opponent's moves. Driving – the environment is partially observable because what's around the corner is not known. **2. Deterministic vs Stochastic:** If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment. A stochastic environment is random in nature and cannot be determined completely by an agent. In a deterministic, fully observable environment, agent does not need to worry about uncertainty. **Eg:** Chess: there would be only a few possible moves for a coin at the current state and these moves can be determined. Self-Driving Cars: the actions of a self-driving car are not unique, it varies time to time. **3. Episodic vs Sequential:** In an Episodic task environment, each of the agent's actions is divided into atomic incidents or episodes. There is no dependency between current and previous incidents. In each incident, an agent receives input from the environment and then performs the corresponding action. **Example:** Consider an example of Pick and Place robot, which is used to detect defective parts from the conveyor belts. Here, every time robot(agent) will make the decision on the current part i.e. there is no dependency between current and previous decisions. In a Sequential environment, the previous decisions can affect all future decisions. The next action of the agent depends on what action he has taken previously and what action he is supposed to take in the future. **Eg:** Checkers- Where the previous move can affect all the following moves.**4. Single-agent vs Multi-agent**: If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment. However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment. The agent design problems in the multi-agent environment are different from single agent environment. **5. Static vs Dynamic:** If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment. Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action. However for dynamic environment, agents need to keep looking at the world at each action. Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment. **6. Discrete vs Continuous:** If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment. A chess gamecomes under discrete environment as there is a finite number of moves that can be performed. A self-driving car is an example of a continuous environment. **7. Known vs Unknown:** Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action. In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action. It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable. **8. Accessible vs Inaccessible:** If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible. An empty room whose state can be defined by its temperature is an example of an accessible environment. Information about an event on earth is an example of Inaccessible environment.

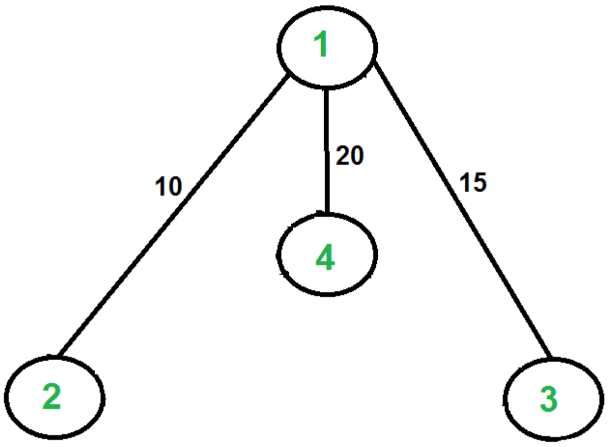
Heuristics are approximations used to minimize the searching process. It is a function that maps from problem state description to measures of desirability, usually represented as number. Which aspect of the problem state are considered, how these aspects are evaluated, and weight given to the individual aspects are chosen. Define a heuristic function h(n) that estimates the "goodness" of a node n. **Specifically, h(n) = estimated cost (or distance) of minimal cost path from n to a goal state.** The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description. Computed in such a way that the value of the heuristic function at a given node in the search give as good estimate as possible of whether that node is on the desired path to a solution**. Two categories of problems use heuristics: 1)** Problems for which no exact algorithms are known and one needs to find an approximate and satisfying solution. E.g., computer vision, speech recognition etc**. 2)** Problems for which exact solutions are known, but computationally infeasible. E.g., Rubik's cube, chess etc. The heuristics which are needed for solving problems are generally represented as a heuristic function which maps the problem states into number. These numbers are then appropriately used to guide search.

In the Blocks World problem, a heuristic function can be defined based on the number of blocks that are out of place or not in their goal positions. **One possible heuristic function for the Blocks World problem is: Count the number of blocks that are not in their goal positions.** This heuristic function estimates the number of blocks that need to be moved to reach the goal state. The goal state is typically defined as having all blocks arranged in the desired positions. To solve the Blocks World problem using this heuristic function, an informed search algorithm such as A* (A-star) can be employed. A* search combines the estimated cost of reaching the goal (heuristic function) with the actual cost of reaching the current state from the initial state (path cost). The algorithm explores the search space by considering the states with the lowest combined cost. **1.** Initialize the search with the initial state of the blocks configuration. **2.** Compute the heuristic value (number of misplaced blocks) for the current state. **3.** Estimate the total cost of reaching the goal state by combining the path cost and the heuristic value. **4.** Expand the current state by generating all possible successor states. **5.** For each successor state, compute its heuristic value and estimate its total cost. **6.** Add the successor states to the search queue based on their total costs, placing states with lower costs at the front. **7.** Repeat steps 2 to 6 until the goal state is reached or the search queue is empty. **8.** If the goal state is reached, a solution path from the initial state to the goal state has been found. Otherwise, there is no solution. By using the heuristic function to guide the search, A* algorithm can efficiently explore the search space and find an optimal or near-optimal solution to the Blocks World problem.



Heuristic function for Travelling Salesman problem: Heuristic functions are commonly used in solving TSP to estimate the cost or desirability of different paths. One commonly used heuristic function for TSP is the **Minimum Spanning Tree (MST) heuristic. The** MST heuristic works by constructing a minimum spanning tree of the cities and then using it to estimate the cost of completing the tour. The steps to solve TSP using the MST heuristic are as follows:**1) Build a complete graph:** Create a complete graph with the cities as nodes and calculate the distances between each pair of cities. **2) Find the Minimum Spanning Tree:** Use a minimum spanning tree algorithm, such as Prim's algorithm or Kruskal's algorithm, to find the minimum spanning tree of the graph. The minimum spanning tree is a tree that connects all the cities with the minimum total edge weight. **3) Double the MST:** Duplicate all the edges of the minimum spanning tree to create a tour that visits each city exactly twice. **4) Find an Eulerian circuit:** Find an Eulerian circuit in the doubled MST. An Eulerian circuit is a path that visits every edge exactly once. **5) Shorten the circuit:** Shorten the Eulerian circuit by eliminating repeated visits to cities. This is done by removing loops and keeping only the first occurrence of each city. **6) Calculate the tour length:** Calculate the total length of the tour by summing the distances between consecutive cities in the shortened circuit. **7) Improve the solution:** Apply optimization techniques, such as 2-opt or 3-opt, to improve the solution by swapping pairs or triplets of edges to reduce the tour length further. **8) Repeat the process:** Repeat steps 1 to 7 multiple times with different initial configurations or heuristics to explore different solutions and find the best one. By using the MST heuristic, the TSP can be solved by estimating the tour length based on the minimum spanning tree, which helps guide the search for an optimal or near-optimal solution.

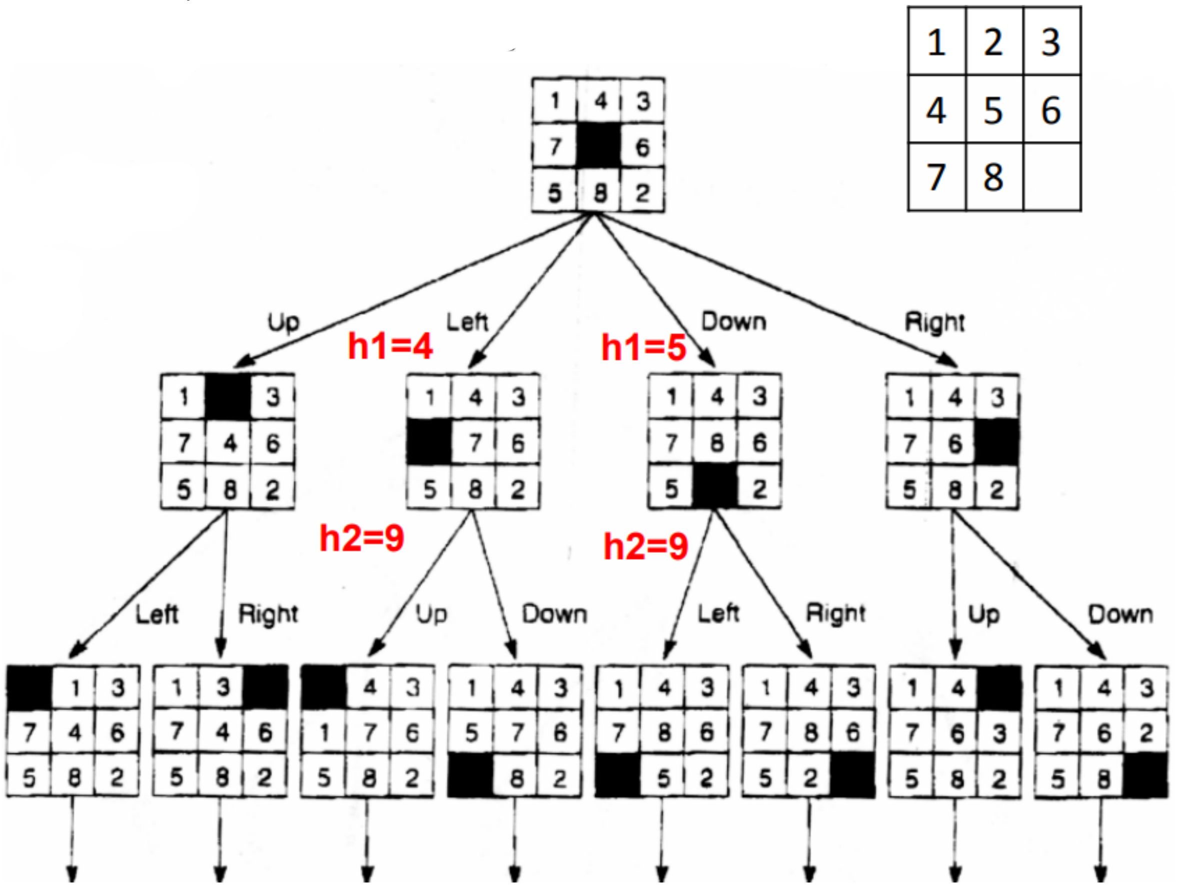




Minimum Spanning Tree of the graph on left side with 1 as root. The Preorder Traversal of MST is 1-2-4-3. So the output is 1-2-4-3-1

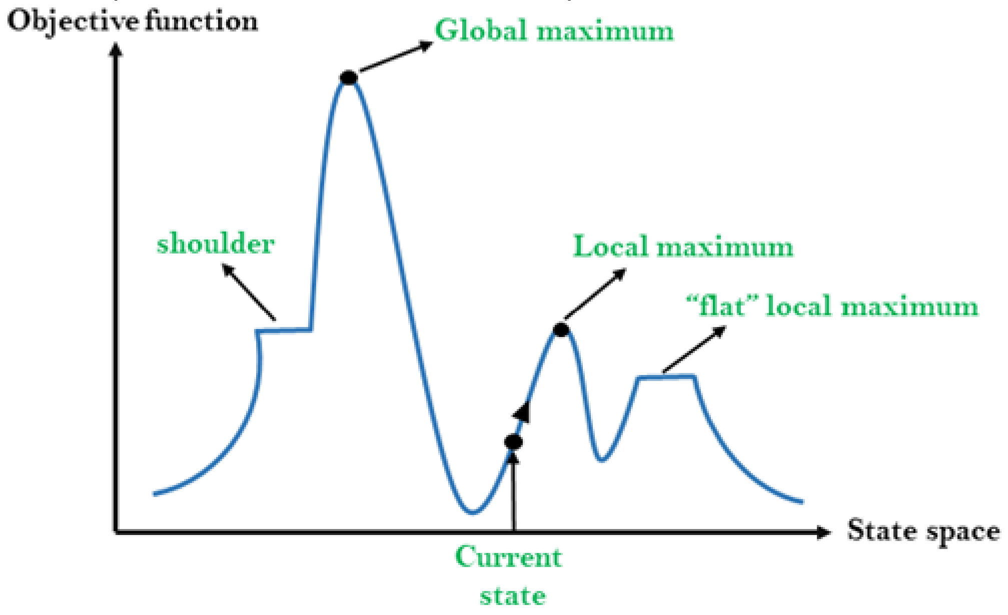
Algorithm: 1) Let 1 be the starting and ending point for salesman. 2) Construct MST from with 1 as root using Prim's Algorithm.3) List vertices visited in preorder walk of the constructed MST and add 1 at the end. **example.** The first diagram is the given graph. The second diagram shows MST constructed with 1 as root. The preorder traversal of MST is 1-2-4-3. Adding 1 at the end gives 1-2-4-3-1 which is the output of this algo. **2-approximate:** The cost of the output produced by the above algorithm is never more than twice the cost of best possible output. Let us see how is this guaranteed by the above algorithm. Let us define a term full walk to understand this. A full walk is lists all vertices when they are first visited in preorder, it also list vertices when they are returned after a subtree is visited in preorder. The full walk of above tree would be 1-2-1-4-1-3-1. Following are some **important facts** that prove the 2-approximateness. **1)** The cost of best possible Travelling Salesman tour is never less than the cost of MST. (The definition of MST says, it is a minimum cost tree that connects all vertices). **2) T**he total cost of full walk is at most twice the cost of MST (Every edge of MST is visited at-most twice) **3)** The output of the above algorithm is less than the cost of full walk. In above algorithm, we print preorder walk as output. In preorder walk, two or more edges of full walk are replaced with a single edge. For example, 2-1 and 1-4 are replaced by 1 edge 2-4. So if the graph follows triangle inequality, then this is always true.

The heuristic function commonly used for the 8-puzzle problem is the Manhattan distance or Manhattan heuristic. It calculates the total distance each tile is away from its goal position by summing the horizontal and vertical distances**. steps:** 1) Start with an initial state of the puzzle. **2)** Compute the heuristic value (h-value) for the initial state using the Manhattan distance heuristic. Add up the distances of each tile from its goal position. **3)** Generate all possible moves from the current state. **4)**For each generated move, calculate the heuristic value using the Manhattan distance heuristic for the resulting state. **5)** Choose the move that leads to the state with the lowest heuristic value. This move will bring you closer to the goal state. **6)** Apply the chosen move to transition to the new state. **7)** Repeat steps 2-6 until you reach the goal state or find the optimal solution. **8)** If the goal state is reached, the puzzle is solved. Otherwise, continue exploring the search space with the lowest heuristic values until a solution is found. The heuristic function guides the search algorithm (such as A* search) by prioritizing states that are closer to the goal state based on the estimated distance. It helps in reducing the search space and finding an optimal solution more efficiently. h1 = number of misplaced tiles; h2 = Manhattan distance



Local search is a heuristic method for solving computationally hard optimisation problems. Local search starts from an initial solution and evolves the single solution that is mostly better solution. At each solution in this path it evaluates a number of moves on the solution and applies the most suitable move to take the step to the next solution. It continues this process for a high number of iterations until it is terminated. Local search uses a single search path and moves facts around to find a good feasible solution. Hence it is natural to implement. Local search algorithms are widely applied to numerous hard computational problems, including problems from artificial intelligence, mathematics, operations research, engineering and bioinformatics. **Some problems where local search is applied are: (1)** The vertex cover problem, in which a solution is a vertex cover of a graph, and the target is to find a solution with a minimal number of nodes. **(2)** The travelling salesman problem, in which a solution is a cycle containing all nodes of the graph and the target is to minimise the total length of the cycle. **(3)** The Hopfield Neural Networks problem for which finding stable configuration in Hop-field network.

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value. It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that. A node of hill climbing algorithm has two components which are state and value. Hill Climbing is mostly used when a good heuristic is available. In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state. **Features: Generate and Test variant: Hill** Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space. **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost. **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.



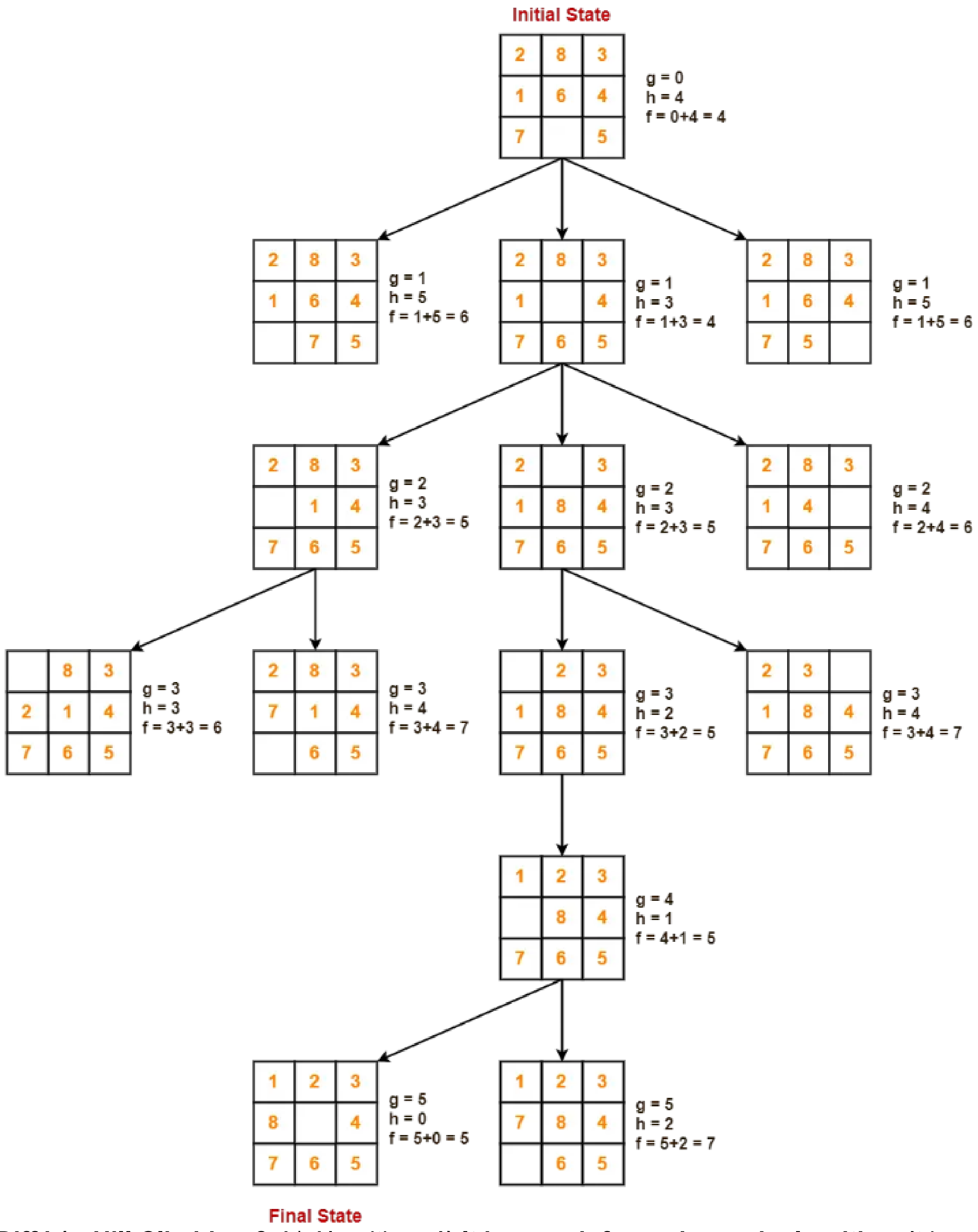
Local Maximum: It is a state which is better than its neighbor states, but there is also another state which is higher than it. **Global Maximum:** It is the best possible state of state space landscape. It has the highest value of objective function. **Current state:** It is a state in a landscape diagram where an agent is currently present. **Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value. **Shoulder:** It is a plateau region which has an uphill edge.
Types: 1) Simple Hill Climbing: It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state. **2) Steepest-Ascent hill climbing:** This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors. **3) Stochastic hill climbing** does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state. **Problems in Hill Climbing Algorithm: 1) Local Maximum:** It is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum. **Solution:** Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well. **2) plateau** is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area. **Solution:** The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region. **3) Ridges:** A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move. **Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.

Simulated annealing is a variation of hill climbing in which, at the beginning of the process, some downhill moves may be made. **1)** The idea is to do enough exploration of the whole space early on so that the final solution is relatively intensive to the starting state. **2)** This should lower the chances of getting caught at a local maximum, a plateau or a ridge.
3) Simulated annealing as a computational process is patterned after the physical process of annealing, in mechanical term Annealing is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state. The same process is used in simulated annealing in which the algorithm picks a random move, instead of picking the best move. **4)** If the random move improves the state, then it follows the same path. **5)** Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path. **6)** Simulated annealing is an effective and general form of optimisation. **7)** Annealing refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. **8)** Simulated annealing uses the objective function of an optimisation problem instead of the energy of a material. **9)** SA is a probabilistic technique for approximating the global optimum of a given function. **10)** Specifically, it is a met heuristic to approximate global optimisation in a large search space for an optimisation problem.

Diff b/w hill climbing and Simulated Annealing: **1) Hill Climbing is a heuristic optimization process that iteratively advances towards a better solution at each step in order to find the best solution in a given search space.** Simulated Annealing is a probabilistic optimization algorithm that simulates the metallurgical annealing process in order to discover the best solution in a given search area by accepting less-than-ideal solutions with a predetermined probability. **2) By iteratively progressing towards a better solution at each stage, Hill Climbing seeks to locate the ideal solution within a predetermined search space.** Simulated annealing seeks the global optimum in a given search space by accepting poorer answers with a predetermined probability. This allows it to bypass local optimum conditions. **3) In order to iteratively move towards the best answer at each stage, Hill Climbing employs a greedy method. It only accepts solutions that are superior to the ones already in place.** Simulated annealing explores the search space and avoids local optimum by employing a probabilistic method to accept a worse solution with a given probability. As the algorithm advances, the likelihood of accepting an inferior answer diminishes. **4) Hill Climbing comes to an end after a certain number of iterations or when it achieves a local optimum.** When the temperature hits a predetermined level or the maximum number of repetitions, simulated annealing comes to an end. **5) Hill climbing is quick and easy, but it has the potential to become locked in local optima and miss the overall best solution.** Simulated annealing is more efficient at locating the global optimum than Hill Climbing, particularly for complicated situations with numerous local optima. Simulated annealing is slower than Hill Climbing. **6) Many different applications, including image processing, machine learning, and gaming, use hill climbing.** Several fields, including logistics, scheduling, and circuit design, use simulated annealing.

The **Minimax algorithm** is a backtracking algorithm used in game theory and decision-making. It is used to find the optimal move for a player, assuming that the opponent is also playing optimally. It is commonly used for turn-based two-player games such as chess, checkers, tic-tac-toe, etc. In minimax, the two players are called minimizer and maximizer. The minimizer tries to get the lowest possible score, while the maximizer attempts to get the maximum possible score. The minimax method determines the best move for MAX, the root node player. The search tree is built by recursively extending all nodes from the root in a depth-first manner until the game ends or the maximum search depth is achieved.
Alpha: It is the best highest value choice that we have found in the path of the maximizer. The starting value of alpha is -∞. **Beta:** It is the best lowest value choice that we have found in the path of the minimizer. The starting value of beta is +∞. The condition for Alpha-beta Pruning is α >= β. Alpha is updated only when it's MAX's time, and Beta can only be updated when it's MIN's turn. The MAX player will only update alpha values, whereas the MIN player will only update beta values. During the reversal of the tree, node values will be transferred to upper nodes instead of alpha and beta values.

A* Algorithm works as: 1) It maintains a tree of paths originating at the start node. 2) It extends those paths one edge at a time. 3) It continues until its termination criterion is satisfied. 4) A* Algorithm extends the path that minimizes the following function: **f(n) = g(n) + h(n);** 'n' is the last node on the path; g(n) is the cost of the path from start node to node 'n'; h(n) is a heuristic function that estimates cost of the cheapest path from node 'n' to the goal node. **eg: 8 puzzle:** Find the most cost-effective path to reach the final state from initial state using A* Algo. **g(n) = Depth of node and h(n) = Number of misplaced tiles.**



Diff b/w Hill Climbing & A* Algorithm**: 1) It is an uninformed search algorithm.** It is an informed search algorithm. **2) It chooses the best available move at each step.** It uses a heuristic function to estimate the cost of reaching the goal. **3) It does not guarantee to find the optimal solution.** It guarantees to find the optimal solution if the heuristic function is admissible and consistent. **4) It may get stuck at local maxima or plateaus.** It avoids getting stuck at local maxima or plateaus by using the heuristic function. **5) It is computationally less expensive.** It is computationally more expensive due to the use of the heuristic function. **6) It is suitable for small search spaces.** It is suitable for large search spaces. **7) It does not require any memory to store the visited nodes**. It requires memory to store the visited nodes and their associated cost values.

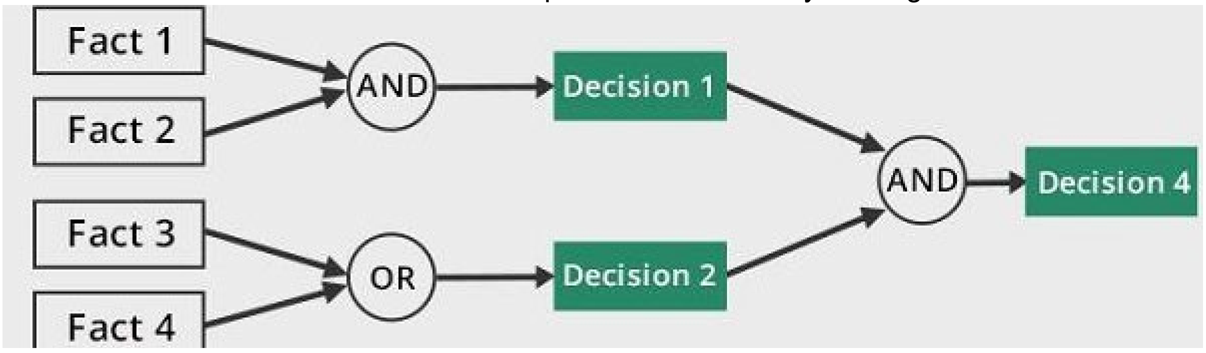
Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. **1) Universal Generalization:** Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).It can be represented as P(c)/ ∀ x P(x) This rule can be used if we want to show that every element has a similar property. In this rule, x must not appear as a free variable. **Example:** Let's represent, P(c): "A byte contains 8 bits", so for ∀ x P(x) "All bytes contain 8 bits.", it will also be true. **2) Universal Instantiation:** It is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences. The new KB is logically equivalent to the previous KB. As per UI, we can infer any sentence obtained by substituting a ground term for the variable. The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse. It can be represented as ∀ x P(x)/ P(c). **Example:** IF "Every person like ice-cream"=> ∀x P(x) so we can infer that "John likes ice-cream" => P(c). **3) Existential Instantiation: It** is also called as Existential Elimination, which is a valid inference rule in first-order logic. It can be applied only once to replace the existential sentence. The new KB is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable. This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c. The restriction with this rule is that c used in the rule must be a new term for which P(c) is true. It can be represented as ∃x P(x)/ P(c). **Example:** From the given sentence: ∃x Crown(x) ∧ OnHead(x, John), So we can infer: Crown(K) ∧ OnHead(K, John), as long as K does not appear in the knowledge base. The above used K is a constant symbol, which is called Skolem constant. The Existential instantiation is a special case of Skolemization process. **4) Existential introduction:** It is also known as an existential generalization, which is a valid inference rule in first-order logic. This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P. It can be represented as: Inference in First-Order Logic Example: Let's say that, “Priyanka got good marks in English." "Therefore, someone got good marks in English." **Generalized Modus Ponens Rule:** For the inference process in FOL, we have a single inference rule which is called Generalized Modus Ponens. It is lifted version of Modus ponens. Generalized Modus Ponens can be summarized as, " P implies Q and P is asserted to be true, therefore Q must be True." According to Modus Ponens, for atomic sentences pi, pi', q. Where there is a substitution θ such that SUBST (θ, pi',) = SUBST(θ, pi), it can be represented as:

$$\frac{p1', p2', \ldots, pn', (p1 \land p2 \land \ldots \land pn \Rightarrow q)}{\mathrm{SUBST}(\theta, q)}$$

Example: We will use this rule for Kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.
Here let say, p1' is king(John); p1 is king(x); p2' is Greedy(y); p2 is Greedy(x)
θ is {x/John, y/John}; q is evil(x) ; SUBST(θ,q).

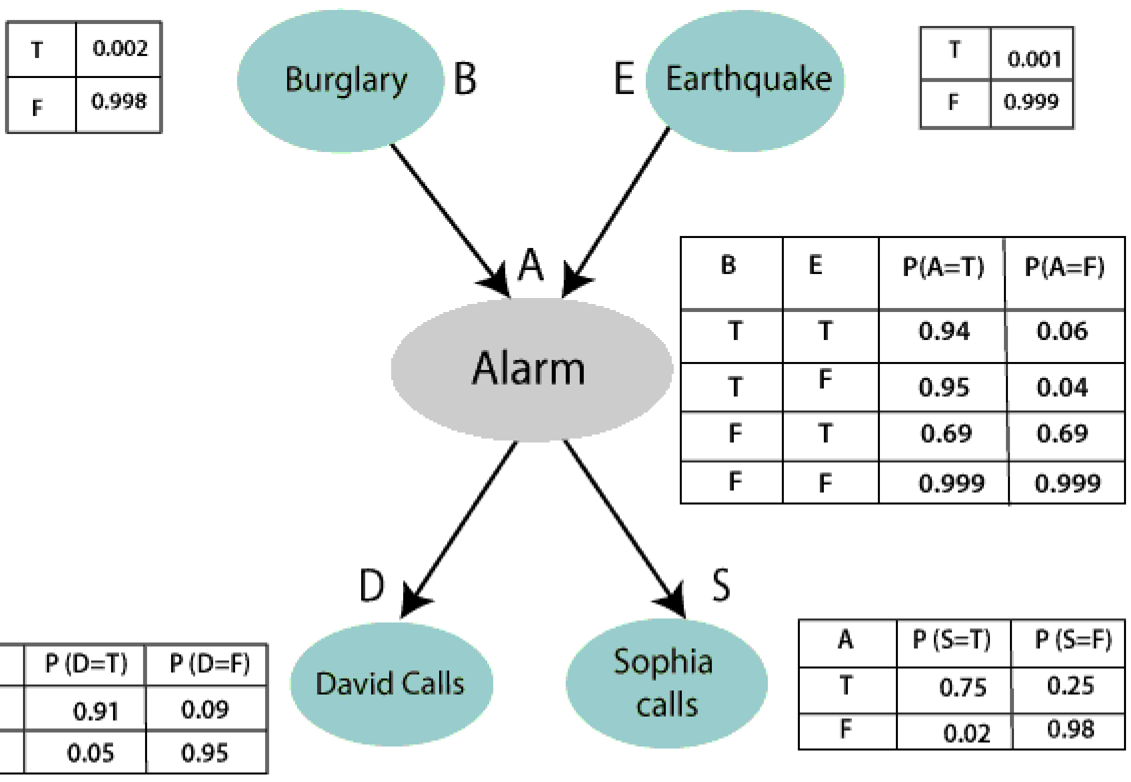
Diff b/w Propositional Logic & Predicate Logic: **1) Propositional logic is the logic that deals with a collection of declarative statements which have a truth value, true or false.** Predicate logic is an expression consisting of variables with a specified domain. It consists of objects, relations and functions between the objects. **2) It is the basic and most widely used logic. Also known as Boolean logic.** It is an extension of propositional logic covering predicates and quantification. **3) A proposition has a specific truth value, either true or false.** A predicate’s truth value depends on the variables’ value. **4) Scope analysis is not done in propositional logic.** Predicate logic helps analyze the scope of the subject over the predicate. There are three quantifiers : Universal Quantifier (∀) depicts for all, Existential Quantifier (∃) depicting there exists some and Uniqueness Quantifier (∃!) depicting exactly one. **5) Propositions are combined with Logical Operators or Logical Connectives like Negation(¬), Disjunction(∨), Conjunction(∧), Exclusive OR(⊕), Implication(⇒), Bi-Conditional or Double Implication(⇔).** Predicate Logic adds by introducing quantifiers to the existing proposition. **6) It is a more generalized representation.** It is a more specialized representation. **7) It cannot deal with sets of entities.** It can deal with set of entities with the help of quantifiers.

Forward chaining is a method of reasoning in artificial intelligence in which inference rules are applied to existing data to extract additional data until an endpoint (goal) is achieved.In this type of chaining, the inference engine starts by evaluating existing facts, derivations, and conditions before deducing new information. An endpoint (goal) is achieved through the manipulation of knowledge that exists in the knowledge base. Forward chaining can be used in planning, monitoring, controling, and interpreting applications. **Properties of forward chaining:** The process uses a down-up approach (bottom to top). It starts from an initial state and uses facts to make a conclusion. This approach is data-driven. It's employed in expert systems and production rule system. Example: A; A->B; B. A is the starting point. A->B represents a fact. This fact is used to achieve a decision B. **eg: DENDRAL** is used in the prediction of the molecular structure of substances. Deducing the chemical structure starts by finding the number of atoms in every molecule. The mass spectrum of the sample is then used to establish the arrangement of the atoms. We can summarize these steps as follows. The chemical formula is determined (the number of atoms in every molecule). The spectrum machine is used to form mass spectrums of the sample. The isomer and structure of the chemical are identified. In this example, the identification of the chemical structure is the endpoint. In the DENDRAL expert system, a generate and test technique is employed. There are two elements in the generator: a synthesiser and structural enumerator. The synthesiser plays the role of producing the mass spectrum. The structural enumerator identifies the structure of substances and prevents redundancy in the generator.



<u>**Switch the arrows other way arnd for**</u> **Backward chaining** is a concept in artificial intelligence that involves backtracking from the endpoint or goal to steps that led to the endpoint. This type of chaining starts from the goal and moves backward to comprehend the steps that were taken to attain this goal. The backtracking process can also enable a person establish logical steps that can be used to find other important solutions. Backward chaining can be used in debugging, diagnostics, and prescription applications. **Properties of backward chaining:** The process uses an up-down approach (top to bottom). It's a goal-driven method of reasoning. The endpoint (goal) is subdivided into sub-goals to prove the truth of facts. A backward chaining algorithm is employed in inference engines, game theories, and complex database systems. The modus ponens inference rule is used as the basis for the backward chaining process. This rule states that if both the conditional statement (p->q) and the antecedent (p) are true, then we can infer the subsequent (q). **Example:** B; A->B; A. B is the goal or endpoint, that is used as the starting point for backward tracking. A is the initial state. A->B is a fact that must be asserted to arrive at the endpoint B. **Eg: The MYCIN** expert system is a real life example of how backward chaining works. This is a system that's used in the diagnosis of bacterial infections. It also recommends suitable treatments for this type of infections. The knowledge base of a MYCIN comprises many antecedent-consequent rules, that enable the system to recognize various causes of (bacterial) infections. This system is suitable for patients who have a bacterial infection, but don't know the specific infection. The system will gather information relating to symptoms and history of the patient. It will then analyze this information to establish the bacterial infection. A suitable sequence can be as follows: The patient has a bacterial infection. The patient is vomiting. He/she is also experiencing diarrhea and severe stomach upset. Therefore, the patient has typhoid (salmonella bacterial infection). The MYCIN expert system uses the information collected from the patient to recommend suitable treatment. The recommended treatment corresponds to the identified bacterial infection. In the case above, the system may recommend the use of ciprofloxacin.

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. **Defined as:** A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph. Also called a Bayes network, belief network, decision network, or Bayesian model. Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection. Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be **used in various tasks including** *prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty*. **Example:** Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry. **Solution: 1)** The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability. **2)** The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling. **3)** The conditional distributions for each node are given as conditional probabilities table or CPT. **4)** Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable. **5)** In CPT, a boolean variable with k boolean parents contains 2K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values. Burglary (B); Earthquake(E); Alarm(A); David Calls(D); Sophia calls(S). P[D, S, A, B, E], can rewrite the above probability statement using joint probability distribution: P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E];
=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E];
= P [D| A]. P [S| A, B, E]. P[A, B, E];
= P[D | A]. P[S | A]. P[A| B, E]. P[B, E];
= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E].



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

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

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

Let's take the observed probability for the Burglary and earthquake component:
P(B= True) = 0.002, which is the probability of burglary.
P(B= False)= 0.998, which is the probability of no burglary.
P(E= True)= 0.001, which is the probability of a minor earthquake
P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

Conditional probability table for Alarm A: depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

The Conditional probability of Sophia that she calls depends on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).
= 0.75* 0.91* 0.001* 0.998*0.999 = 0.00068045.

The inference that can be drawn from this example is that we can use the Bayesian network to compute the posterior probability of any variable given some evidence. For example, we can compute the probability of a burglar given that John calls and Mary does not call using Bayes' rule and marginalization. This can help us make decisions under uncertainty based on the available data and our prior knowledge.

The process of conditioning is called as probability propagation **or inference** or belief updating. This is performed via a "flow of information" through the network, which is not limited to the directions the arcs inference The inference in the probabilistic system is nothing but computing the posterior probability distribution a set of query node when some evidence nodes values are specified.

Conditional independence is a property that describes the relationship between variables in a belief network. Two variables A and B are conditionally independent given a third variable C if the probability distribution of A is independent of B, given the value of C. **Eg:** let's consider a belief network with variables A, B, and C. If A and B are conditionally independent given C, it means that the probability distribution of A does not change based on the values of B, given the value of C. In other words, knowing the value of C makes the relationship between A and B independent of each other.

Conditional probability is a fundamental concept in AI that measures the likelihood of an event occurring given the occurrence of another event. It is denoted as P(A | B), which represents the probability of event A happening given that event B has already occurred. Conditional probability plays a crucial role in various AI techniques, including Bayesian Networks, decision trees, and machine learning algorithms. It allows us to model dependencies and relationships between variables, make predictions based on observed evidence, and update probabilities as new information becomes available. In AI, conditional probability is used for tasks such as classification, prediction, anomaly detection, and decision-making. It helps in reasoning under uncertainty, making probabilistic inferences, and handling incomplete or noisy data.

Bayes' theorem: P(A|B)= [P(B|A) P(A)] / P(B)determines the probability of an event with uncertain knowledge. **P(A|B)**: **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B. **P(B|A): likelihood**, in which we consider that hypothesis is true, then we calculate the probability of evidence. **P(A):prior probability,** probability of hypothesis before considering the evidence. **P(B):marginal probability,** pure probability of an evidence. **Application:** It is used to calculate the next step of the robot when the already executed step is given. Bayes' theorem is helpful in weather forecasting. It can solve the Monty Hall problem.

Uncertainty in AI refers to the lack of complete knowledge or information about a situation or event. It is a fundamental aspect of real-world problems, where there is often ambiguity, variability, and unpredictability. AI systems deal with uncertainty by employing various techniques to model and reason under uncertain conditions. Types of uncertainties:
1. Epistemic Uncertainty: This type of uncertainty arises due to incomplete knowledge or information about a system or problem. It reflects our limited understanding of the world and can be reduced with more data or improved models. **2. Aleatoric Uncertainty:** Aleatoric uncertainty is inherent to the system or environment itself and cannot be reduced even with additional data or knowledge. It represents the inherent randomness or variability of events.
AI techniques and models that address uncertainty include: 1) Probabilistic Models: Probabilistic models, such as Bayesian Networks and Markov Decision Processes, explicitly represent uncertainty using probability distributions. They provide a framework for reasoning under uncertainty and enable probabilistic inference and decision-making. **2) Fuzzy Logic:** Fuzzy logic allows for the representation of uncertain or vague concepts by assigning degrees of membership to different categories. It is useful when dealing with imprecise or subjective information. **3) Monte Carlo Methods:** Monte Carlo methods use random sampling techniques to estimate probabilities or analyze complex systems under uncertainty. They simulate multiple possible outcomes to obtain statistical approximations. **4) Bayesian Inference:** Bayesian inference is a statistical technique that updates beliefs or probabilities based on observed evidence. It combines prior knowledge with observed data to obtain posterior probabilities and make informed decisions.

Resolution: It is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form. **Clause:** Disjunction of literals (an atomic sentence) is called a clause. It is also known as a unit clause. **Conjunctive Normal Form:** A sentence represented as a conjunction of clauses is said to be CNF. **Steps for Resolution:**
1. Conversion of facts into first-order logic. **2.** Convert FOL statements into CNF
1) Eliminate all implication (→) and rewrite 2) standardize variables 3) Move negation (¬) inwards and rewrite 4) Skolemization (Eliminate existential instantiation quantifier by elimination) 5) Drop Universal quantifiers
3. Negate the statement which needs to prove (proof by contradiction) **4.** Draw resolution graph (unification).

Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process. It takes two literals as input and makes them identical using substitution. Let Ψ1 and Ψ2 be two atomic sentences and σ be a unifier such that, $\Psi1\sigma = \Psi2\sigma$, then it can be expressed as UNIFY(Ψ1, Ψ2).
conditions for unification: 1) Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified. 2) Number of Arguments in both expressions must be identical. 3) Unification will fail if there are two similar variables present in the same expression. **Example:** Find the MGU for Unify{King(x), King(John)}
Let Ψ1 = King(x), Ψ2 = King(John), Substitution θ = {John/x} is a unifier for these atoms and applying this substitution, and both expressions will be identical.The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist). Unification is a key component of all first-order inference algorithms. It returns fail if the expressions do not match with each other. The substitution variables are called Most General Unifier or MGU.

Supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data**. For instance**, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this: a) If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –Apple. B) If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –Banana.
Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it. Since the machine has already learned the things from previous data and this time has to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit). **Two categories of algorithms:**
Classification: A classification problem is when the output variable is a category, such as "Red" or "blue" , "disease" or "no disease". **Regression:** A regression problem is when the output variable is a real value, such as "dollars" or "weight".
Supervised learning deals with or learns with "labeled" data. This implies that some data is already tagged with the correct answer. **Types:** Regression, Logistic Regression, Classification, Naive Bayes Classifiers, K-NN, Decision Trees, Support Vector Machine
Advantages: 1)Supervised learning allows collecting data and produces data output from previous experiences.2) Helps to optimize performance criteria with the help of experience. 3) Supervised machine learning helps to solve various types of real-world computation problems. 4) It performs classification and regression tasks. 5) We have complete control over choosing the number of classes we want in the training data. **Disadvantages:** 1)Classifying big data can be challenging. 2) Training for supervised learning needs a lot of computation time. So, it requires a lot of time. 3) Supervised learning cannot handle all complex tasks in Machine Learning. 4) Computation time is vast for supervised learning. 5) It requires a labelled data set.

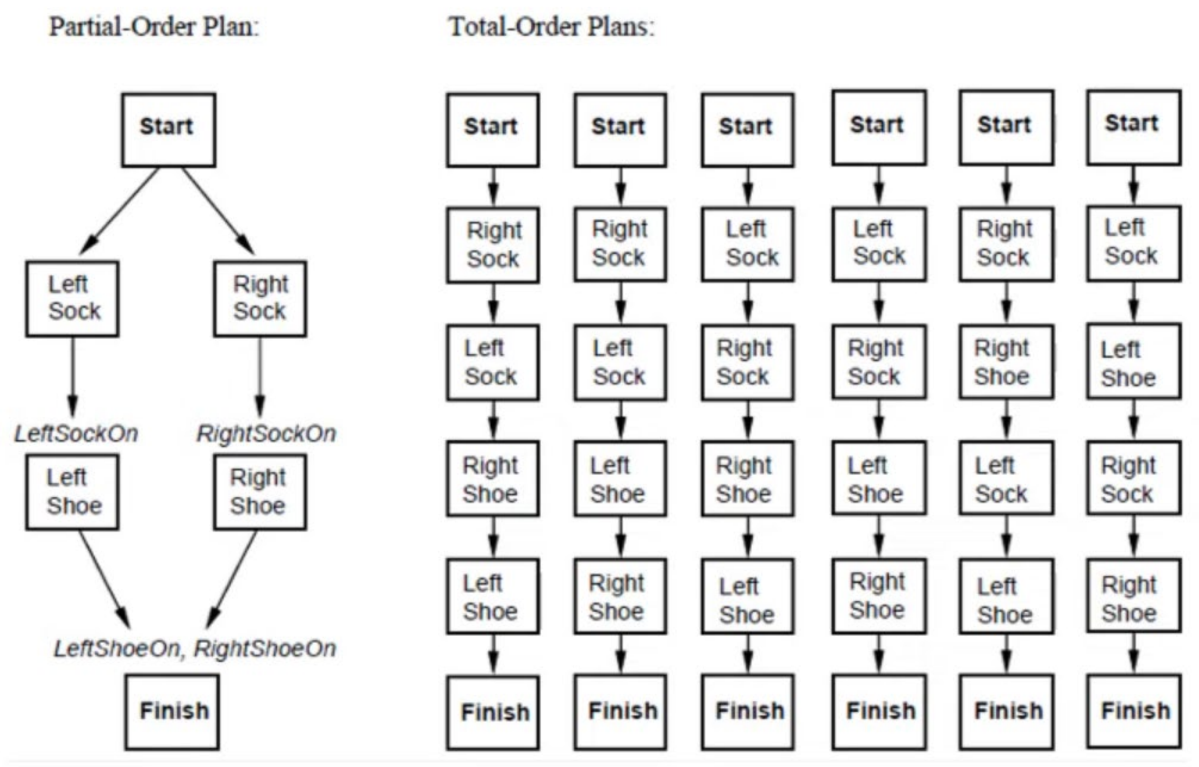
Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself. **For instance, suppose** it is given an image having both dogs and cats which it has never seen. Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats '. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having dogs in them and the second part may contain all pics having cats in them. Here you didn't learn anything before, which means no training data or examples. It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data. **Two categories of algorithms:**
Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior. **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Types: Exclusive (partitioning); Agglomerative; Overlapping; Probabilistic. **Clustering Types:** Hierarchical clustering; K-means clustering; Principal Component Analysis; Singular Value Decomposition; Independent Component Analysis**. Advantages:** 1) Unsupervised learning is flexible in that it can be applied to a wide variety of problems, including clustering, anomaly detection, and association rule mining. 2) Unsupervised learning allows for the exploration of data and the discovery of novel and potentially useful patterns that may not be apparent from the outset. 3) Unsupervised learning is often less expensive than supervised learning because it doesn't require labeled data, which can be time-consuming and costly to obtain.

Disadvantages: 1) Difficult to measure accuracy or effectiveness due to lack of predefined answers during training. **2)** The results often have lesser accuracy**. 3)** The user needs to spend time interpreting and label the classes which follow that classification. **4) Lack of guidance:** Unsupervised learning lacks the guidance and feedback provided by labeled data, which can make it difficult to know whether the discovered patterns are relevant or useful. **5) Sensitivity to data quality**: Unsupervised learning can be sensitive to data quality, including missing values, outliers, and noisy data. **6) Scalability:** Unsupervised learning can be computationally expensive, particularly for large datasets or complex algorithms, which can limit its scalability.

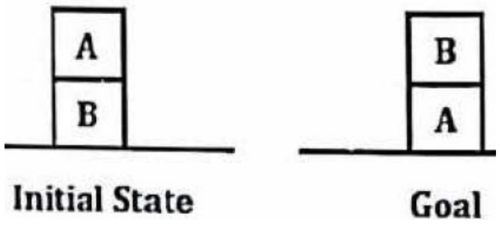
Reinforcement learning 1) It is a type of machine learning that focuses on an agent learning to interact with an environment to maximize its cumulative reward. **2)** The agent learns through trial and error by taking actions in the environment, receiving feedback in the form of rewards or punishments. **3)** The goal of reinforcement learning is to find an optimal policy that maximizes the long-term expected reward. **4)** Key components of reinforcement learning include the agent, the environment, actions, rewards, and the learning algorithm. **5)** The agent observes the state of the environment, selects actions based on its policy, and receives feedback in the form of rewards. **6)** The learning algorithm uses this feedback to update the agent's policy over time, improving its decision-making. **7)** Reinforcement learning can be categorized into value-based methods, policy-based methods, and actor-critic methods. **8)** Value-based methods, such as Q-learning, learn the value of each state-action pair to make optimal decisions. **9)** Policy-based methods directly learn a policy that maps states to actions without explicitly learning values. **10)** Actor-critic methods combine value-based and policy-based approaches, using a value function and a policy function to guide learning. **11)** For example, in the game of chess, a reinforcement learning agent can learn to make moves that lead to winning positions based on rewards and punishments received during gameplay. **12)** Another example is training a robot to navigate through a maze, where the robot learns to take actions that lead to a goal state while avoiding obstacles based on feedback received from the environment.

Planning in AI is about decision-making actions performed by robots or computer programs to achieve a specific goal. Execution of the plan is about choosing a sequence of tasks with a high probability of accomplishing a specific task. **For example,** Planning is required to reach a particular destination. It is necessary to find the best route in Planning, but the tasks to be done at a particular time and why they are done are also very important. That is why Planning is considered the logical side of acting. In other words, Planning is about deciding the tasks to be performed by the artificial intelligence system and the system's functioning under domain-independent conditions. We require domain description, task specification, and goal description for any planning system. A plan is considered a sequence of actions, and each action has its preconditions that must be satisfied before it can act and some effects that can be positive or negative. 2 types: **1) Forward State Space Planning (FSSP):** behaves in the same way as forwarding state-space search. It says that given an initial state S in any domain, we perform some necessary actions and obtain a new state S' (which also contains some new terms), called a progression. It continues until we reach the target position. Action should be taken in this matter. **Disadvantage:** Large branching factor. A**dvantage:** The algorithm is Sound. **2) Backward State Space Planning (BSSP):** behaves similarly to backward state-space search. In this, we move from the target state g to the sub-goal g, tracing the previous action to achieve that goal. This process is called regression (going back to the previous goal or sub-goal). These sub-goals should also be checked for consistency. The action should be relevant in this case. **Disadvantages:** not sound algorithm (sometimes inconsistency can be found). **Advantage:** Small branching factor.

Partial Order Planning: Progression and regression planning are totally ordered plan search forms. They cannot take advantage of problem decomposition. Decisions must be made on how to sequence actions on all the subproblems. Least commitment strategy: Delay choice during search. Partially ordered collection of steps with **a)**Start step has the initial state description as its effect **b)** Finish step has the goal description as its precondition causal links from outcome of one step to precondition of another c) temporal ordering between pairs of steps. Open condition precondition of a step not yet causally linked. A plan is complete iff every precondition is achieved. A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it. Partially Ordered Plans: no strict sequence, partly parallel, observe threats. Shoe and sock example: Any planning algorithm that can place two actions into a plan without which comes first is a PO plan.



Partial Order Planner: Planning is the process of thinking about and organizing the activities required to achieve a desired goal. Any planning algorithm that can place two actions into a plan without specifying which comes first is called as a Partial Order Planner. A Partial Order Planner searches through plane space. It starts with an initial plan representing the start and finishing steps, and on each iteration adds one more step. If it leads to incomplete plan, it backtracks and tries another branch of search space. The solution is represented as a graph of actions, not a sequence of actions. **EXAMPLE:** Let us define following two macros for the sake of simplicity for block world example.

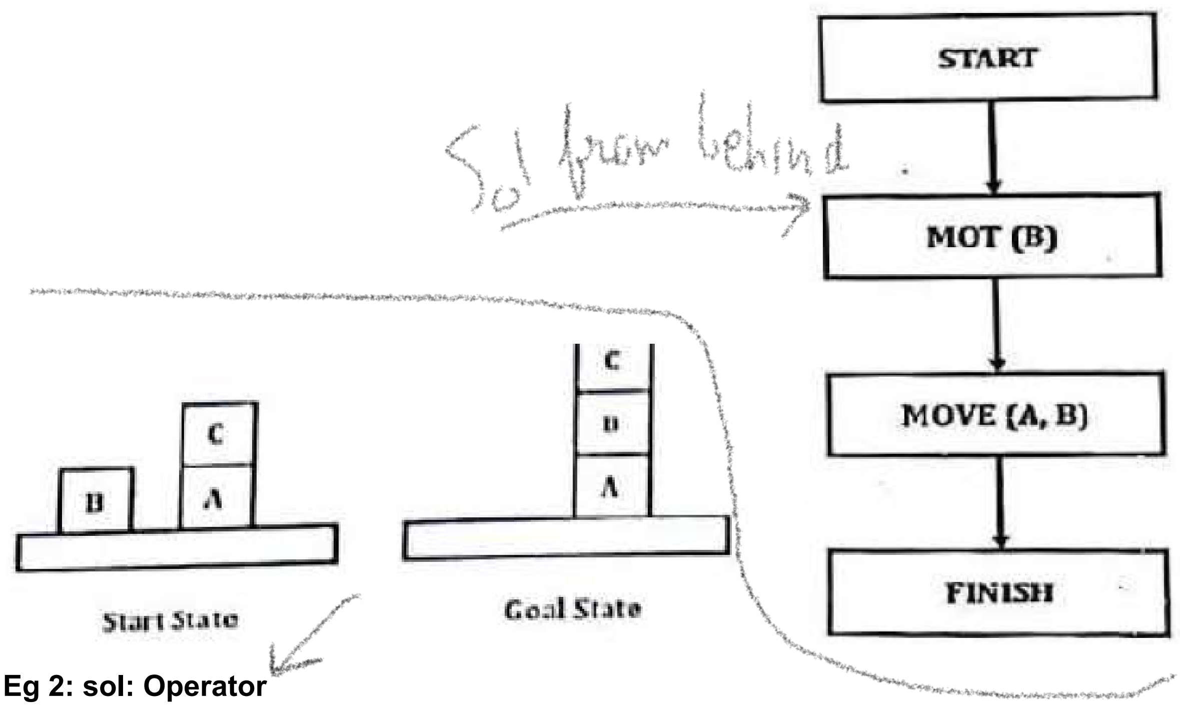


Macro operator	**Description**
MOT (A)	Move A onto table
MOVE (B, A)	MOVE B onto A

To achieve the Goal State ON (A, B): 1) Move 'A' onto table should occur before move 'B' to ‘A’ **2)** Hence MOT (A) should come before MOVE (B, A) in the Final Plan.

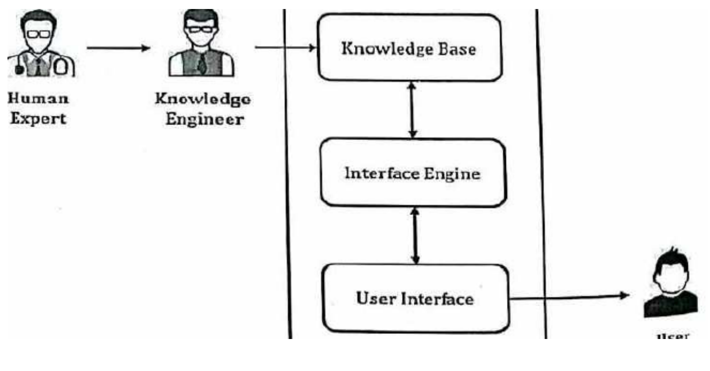
PARTIAL ORDER PLANNER GRAPH: 1. It contains the dummy actions START & FINISH to mark the beginning and end of the plan in the graph. **2.** The planner can generate total plans from the graph.

Diagram behind:



Eg 2: sol: Operator
Unstack(C,A); Putdown(C); Pickup(A); Stack(B,A); Pickup(C); Stack(C,B)

Expert System Arch: Expert System are Artificial Intelligence (AI) tools. It capture the expertise of knowledge workers (Experts) and provide advice to usually non experts in a given domain. In AI, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning about knowledge, represented primarily as if-then rules rather than through conventional procedural code. Expert System are implemented with artificial Intelligence technology, often called Expert System Shells. **Characteristics:** High performance; Understandable; Reliable; Highly responsive. **Components: 1)** Knowledge Base: Knowledge Base is database of rules. It contains domain-specific and high-quality knowledge. Knowledge is required to exhibit intelligence. The success Of any Expert System majorly depends upon the collection of highly accurate and precise knowledge. **2) Interface Engine:** Interface Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular-solution. Interface Engine use Forward & Backward Chaining Strategies. **3) User Interface:** User interface provides interaction between user of the Expert System and the Expert System itself. It is generally Natural Language Processing. so as to be used by the user who is well-versed in the task domain. The user of the Expert System need not be necessarily an expert in Artificial Intelligence. It explains how the Expert System has arrived at a particular recommendation. **Advantages: 1) Availability:** They are easily available due to mass production of software. **2) Less Production Cost:** Production cost is reasonable. This makes them affordable. **3) Speed:** They offer great speed. They reduce the amount of work an individual puts in. **4) Less Error Rate:** Error rate is low as compared to human errors. **5) Reducing Risk:** They can work in the environment dangerous to humans. **6) Steady Response:** They work steadily without getting motional, tensed or fatigued. **Disadvantages: 1)** Limitations ofthe technology. 2) Difficult knowledge acquisition 3) ES are difficult to maintain. 4) High development costs.
Applications: 1) Design Domain 2) Medical Domain 3) Knowledge Domain 4) Finance;



Natural Language Processing (NLP) refers to AI method of communicating with an intelligent systems using a natural language such as English. Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc. The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be Speech or Written Text. **NLP Terminology: 1)Phonology:** It is study of organizing sound systematically. **2)Morphology**: It is a study of construction of words from primitive meaningful units. **3)Morpheme**: It is primitive unit of meaning in a language. **4)Syntax:** It refers to arranging words to make a sentence. It also involves determining the structural role of words in the sentence and in phrases. **5)Semantics:** It is concerned with the meaning of words and how to combine words into meaningful phrases and sentences. **6)Pragmatics:** It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected. **7)Discourse:** It deals with how the immediately preceding sentence can affect the interpretation of the next sentence. **8)World Knowledge:** It includes the general knowledge about the world.
Components of NLP: 1) Natural Language Understanding (NLU): Understanding involves the following tasks: Mapping the given input in natural language into useful representations. Analyzing different aspects of the language. **2) Natural Language Generation (NLG): It** is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation. It involves: **Text planning −** It includes retrieving the relevant content from knowledge base. **Sentence planning −** It includes choosing required words, forming meaningful phrases, setting tone of the sentence. **Text Realization −** It is mapping sentence plan into sentence structure.

Steps in NLP: 1) Lexical Analysis − It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words.
2) Syntactic Analysis (Parsing) − It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.
3) Semantic Analysis − It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”. **4) Discourse Integration −** The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence. **5) Pragmatic Analysis −** During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.
Implementation aspects of Syntactic Analysis: 1)Context-Free Grammar 2)Top-Down Parser
eg of 1) “The bird pecks the grains”
Articles (DET) − a | an | the; Nouns − bird | birds | grain | grains
Noun Phrase (NP) − Article + Noun | Article + Adjective + Noun = DET N | DET ADJ N
Verbs − pecks | pecking | pecked; Verb Phrase (VP) − NP V | V NP;
Adjectives (ADJ) − beautiful | small | chirping
The parse tree breaks down the sentence into structured parts so that the computer can easily understand and process it. In order for the parsing algorithm to construct this parse tree, a set of rewrite rules, which describe what tree structures are legal, need to be constructed. These rules say that a certain symbol may be expanded in the tree by a sequence of other symbols. According to first order logic rule, if there are two strings Noun Phrase (NP) and Verb Phrase (VP), then the string combined by NP followed by VP is a sentence. **The rewrite rules: S → NP VP; NP → DET N | DET ADJ N; VP → V NP**
Lexocon: DET → a | the; ADJ → beautiful | perching; N → bird | birds | grain | grains
V → peck | pecks | pecking
DisADV: They are not highly precise. For example, “The grains peck the bird”, is a syntactically correct according to parser, but even if it makes no sense, parser takes it as a correct sentence.