EXPERIMENT NO: 02

Aim: Identify suitable Agent Architecture for the problem.

Theory: For a Planning Agent the most suitable agent architecture is typically a Goal-based Agent or a Problem-Solving Agent, which focuses on selecting actions to achieve specific goals.

1 Goal-Based Agent Architecture:

*Components:

- Perception: The agent observes the current state of the environment
- Goal Generator: Defines the goals the agent aims to achieve
- Planner: Generates a sequence of actions based on the current state and goal.
- Action Executor: Executes the planned actions in the environment.
- State & Goal Monitor: Continuously checks if the goal has been achieved and if the state is as expected.

2. Model-Based Reflex Agent Architectures

* Components:

- Sensors: Gather information about the environment

- Internal Model: Maintains a model of the world that tracks the agent's current state.
- Actuators: Executes actions to change the state of the environment.

5. Hierarchical Agent Architecture:

* Components:

- High-level planner: Responsible for defining long-term goals and sub-goals.
- Low-level Controller: Handles the execution of specific tasks or action
- Hierarchial structure: Breaks down complex task into simpler, smaller tasks.

Conclusion: The most suitable architecture for a Planning Agent is typically a Goal-based Agent, which focuses on selecting actions to achieve specific goals.

EXPERIMENT NO: 03

AIM: Simple programs using PROLOG as an AI programming language.

THEORY:

Prolog is a logic programming language. It has important role in artificial intelligence. Unlike many other programming languages, Prolog is intended ~~many other~~ primarily as a declarative programming language. Prolog is intended primarily as a declarative programming language.

In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the logic being applied. Formulation or computation is ~~called~~ carried out by running a query over these relations.

KEY FEATURES:

1) Unification: The basis idea is, can the given terms be made to represent the same structure.

2) Backtracking: When a task fails, prolog traces backwards and tries to satisfy previous task.

3) Recursion: Recursion is the basis for any search in program.

SYNTAX:

Format: relation (entity1, entity2, ... k'th entity)

Example: friends (raju, mahesh)
singer (sonu)
odd_number (5)

Explaination : These facts can be interpreted as:
raju and mahesh are friends.
sonu is a singer
5 is a odd number.

ADVANTAGES:

1) Easy to build database. Doesn't need a lot of programming effort.

2) Pattern matching in list handling. Makes it easier to play with any algorithm involving lists.

DISADVANTAGES:

1) LISP (another logic programming language) dominates over prolog with respect to I/O features.

2) Sometimes input and output is not easy.

CONCLUSION: Prolog has unique capabilities that make it suitable for various AI applications, especially in: Expert systems: Prolog can represent knowledge and reason logically making it ideal for building expert systems that can capture and apply the knowledge of human expert in specific domains.

EXPERIMENT NO. 4

AIM: Implement any one of the Uninformed search techniques.

THEORY:

Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algo do not have additional information about state or search space other than how to tranverse the tree, so it is also called blind search.

Types of uninformed search algorithms:

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

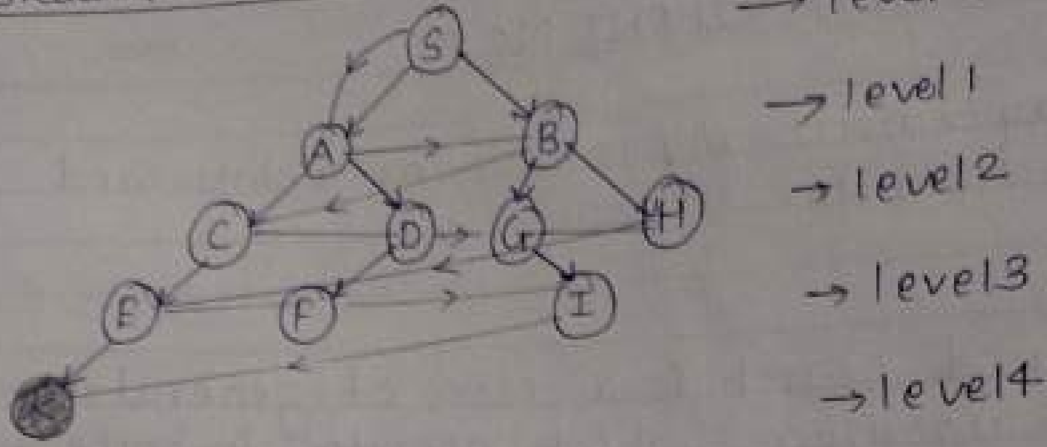
Advantages: ① BFS will provide a solution if any solution exists

② BFS will provide the minimal solution which requires the least number of steps.

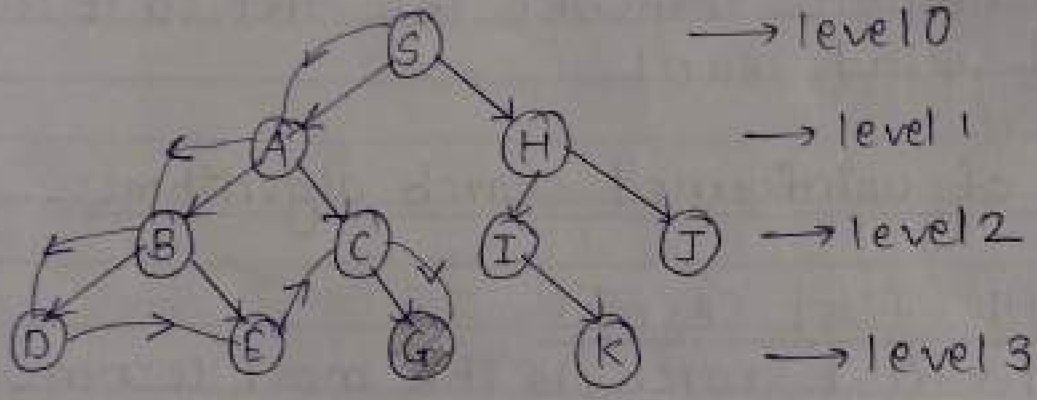
Disadvantages: ① It requires lots of memory to expand the next level.

② BFS needs lots of time if the solution is far away from the root node.

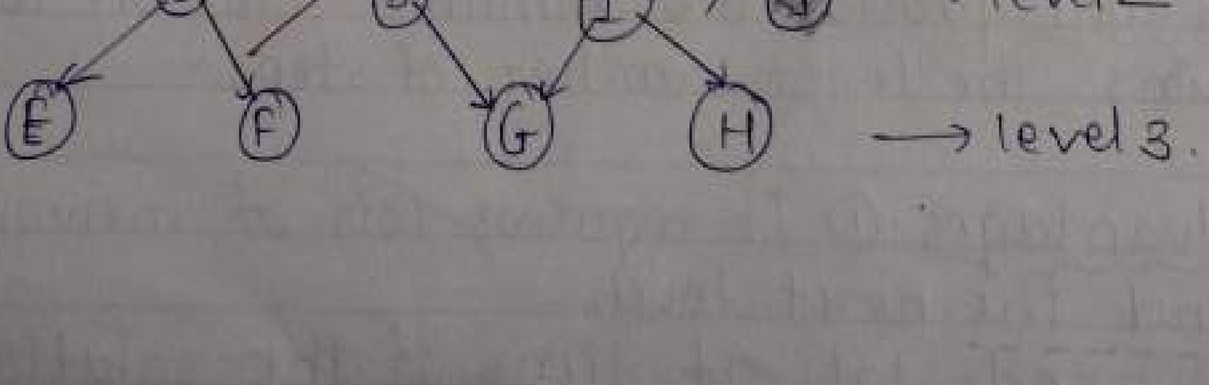
- **Breadth first search**



- **Depth first search**



- **Depth limited Search**



2) Depth-first search:

- Depth-first search is a 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.

Advantage: ① DFS requires very less memory

Disadvantage: ② DFS algorithm goes for deep down searching and sometime it may go to the infinte loop.

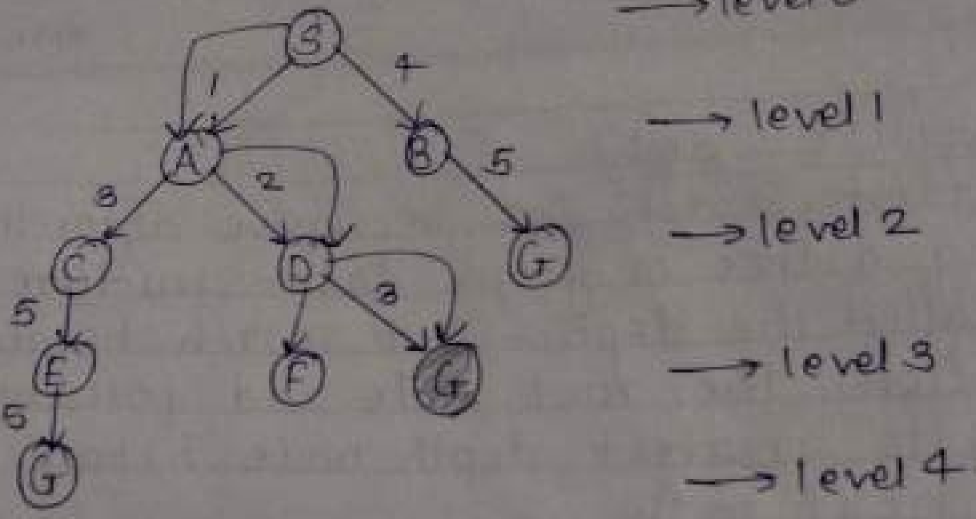
3) Depth-Limited search Algorithm:

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

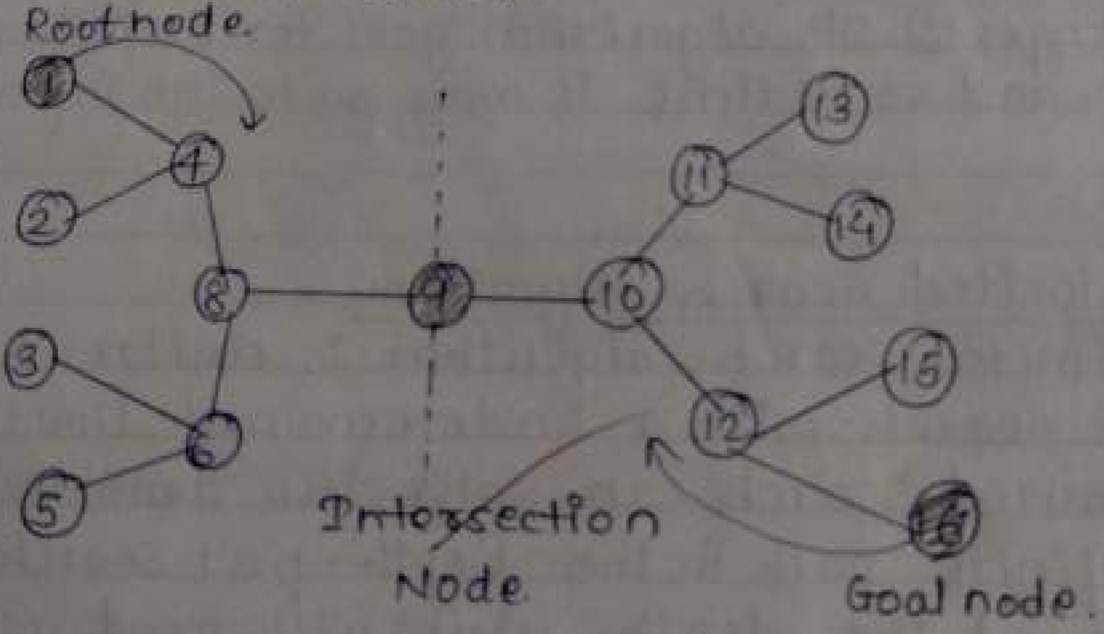
Advantages: ① Depth-limited search is Memory efficient

Disadvantages: ① It also has a incompleteness.
② It may not be optimal if the problem has more than one solution

- Uniform Cost Search.



- Bi directional Search.



4) 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.

Advantages: ① uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages: ① algorithm may be stuck in infinite loop.

5) Bidirectional Search Algorithm:

- Bidirectional search algorith runs two simultaneous searches, one form initial state called as forward search & other from goal node called as backward search to find the goal node.

Advantages: ① Bidirectional search is fast

② Bidirectional search requires less memory

Disadvantages: ① Implementation of the bidirectional search tree is difficult.

② One should know the goal state in advance.

CONCLUSION: This is the complete analysis of all the Uninformed Search Strategies. Each search algorithm is no less than the other.

Page No.
Date

EXPERIMENT NO: 5

AIM: Implement any one of the Informed search techniques

THEORY:

- In an informed search algorithm that combine the cost to reach a state (g-value) with a heuristic estimate of the cost to reach the goal (h-value).
- It is widely used for solving problems like the 8-puzzle, as it finds optimal solutions efficiently.
- It relies on an admissible heuristic to guide the search.

- The 8-puzzle is a well-known problem in the field of AI and puzzle-sloving. It serves as an intriguing model problem with various applications and is widely used to explore heuristic search and state-space exploration.

DESCRIBE: The 8-Puzzle state-

The state of the 8-puzzle is represented using a 3x3 grid, where each cell can hold one of the numbered tiles or remain empty (occupied by blank title). This grid serves as a compact and systematic way to capture the configuration of the puzzle.

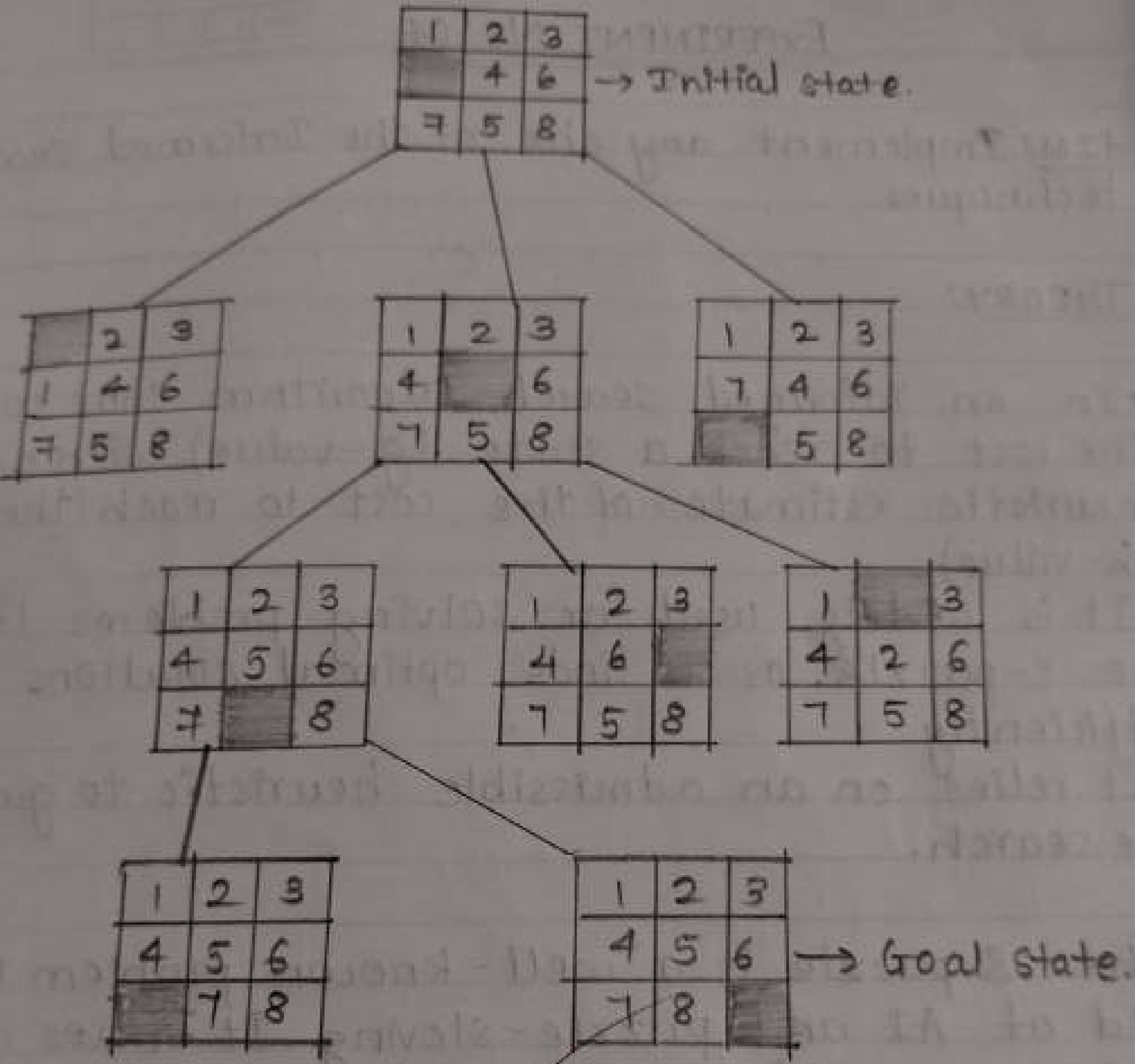


fig: 8 puzzble problem.

INITIAL & GOAL STATES:

In the context of the 8-puzzle, two fundamental states are of particular importance: the initial state and the goal state.

1) Initial State:

- The initial state of the 8-puzzle represents the starting configuration. It's the state from which the puzzle-solving process begins

- The problem-solving algorithm aims to transform the initial state into the goal state using a sequence of valid moves.

2) Goal state:

- The goal state represents the desired configuration that the puzzle should reach.

- In most cases, the goal state involves arranging the numbered tiles in ascending order from left to right and top to bottom, with the blank tile in the bottom right corner.

- Achieving the goal state demonstrates the successful solution of the puzzle.

CONCLUSION: We conclude that rearranging the tiles so that they are in row major order, using as few moves as possible

EXPERIMENT NO: 6

AIM: Implement adversial search using min-max algorithm

THEORY:

- Minmax algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Min-max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI Such as chess, checkers, tic-tac-toe and various tow-players game
- In this algorithm two players play the game, one is called MAX and other is called MIN.

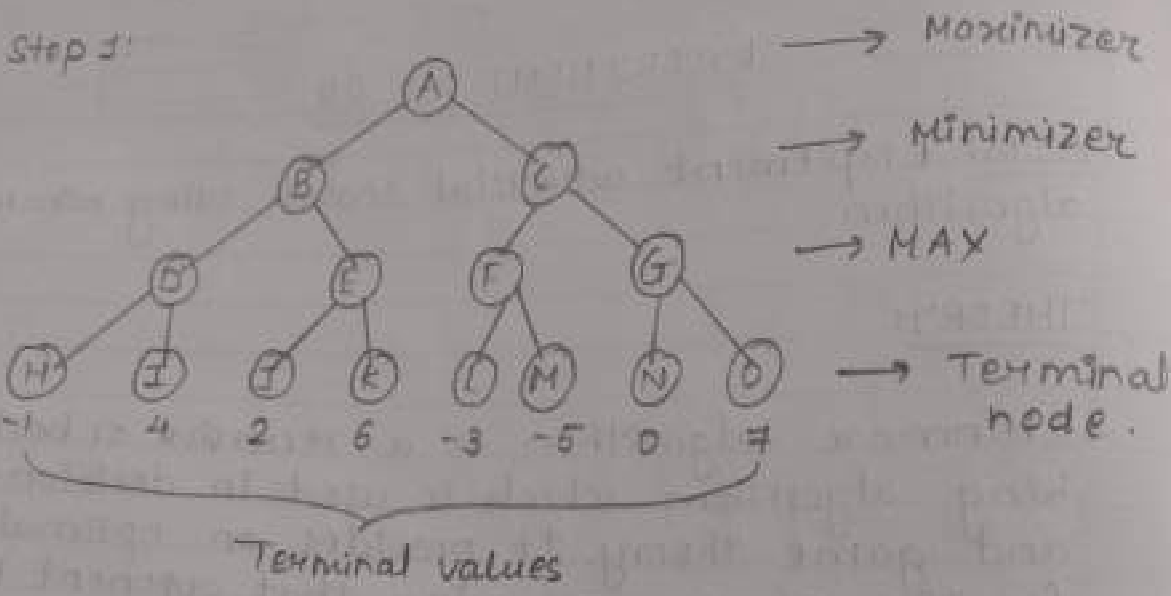
Properties of Mini-Max algorithm:

1) Complete - Min-Max algorithm is complete. It will definitely find a solution (if exist), in the finite search tree.

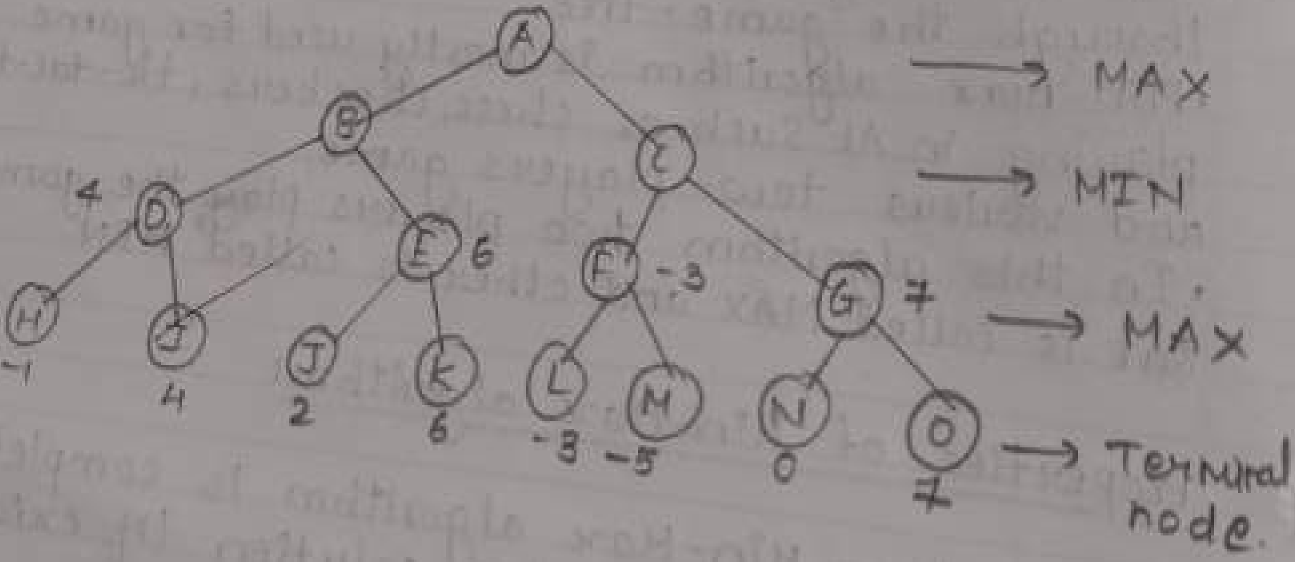
2) Optimal - Min-Max algorithm is optimal if both opponents are playing optimally.

3) Time complexity - As it perform DFS for the game-tree so the time complexity of Min-Max

Step 1:


Maximizer
Minimizer
MAX
Terminal node.
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
-1
4
2
6
-3
-5
0
7
Terminal values

Step 2:


MAX
MIN
MAX
Terminal node.
A
B
C
4
D
E
6
F
-3
G
7
H
I
J
K
L
M
N
O
-1
4
2
6
-3
-5
0
7

algorithm is $O(b^m)$, where b is branching factor of the game-tree, and m is the maximum depth of the tree

1) Space Complexity - Space complexity of Mini-max algorithm is also similar to DFS which is $O(bm)$.

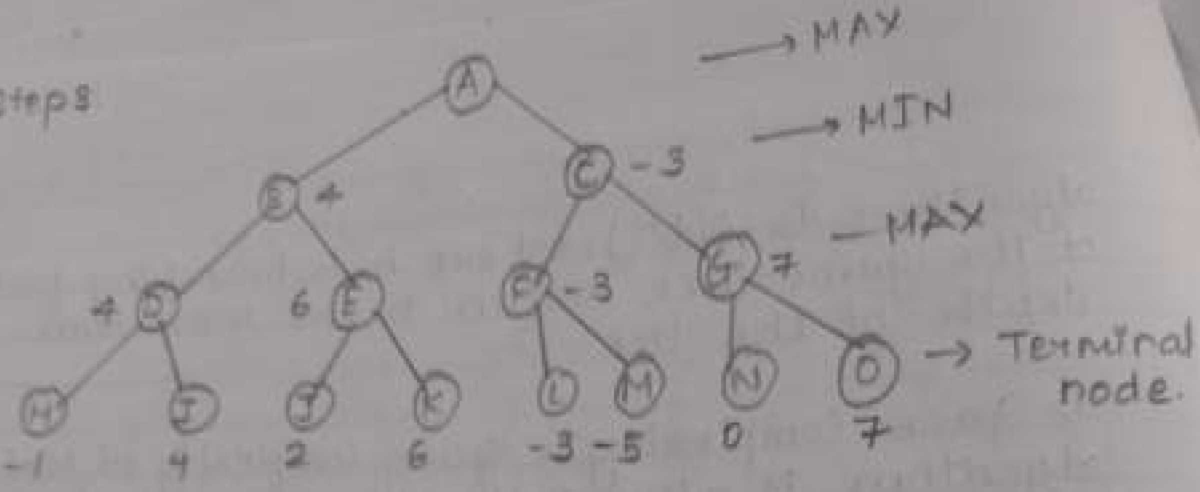
Working of Min-Max Algorithm:

Step 1: 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 = $-\infty$, and minimizer will take next turn which has worst-case initial value = $+\infty$.

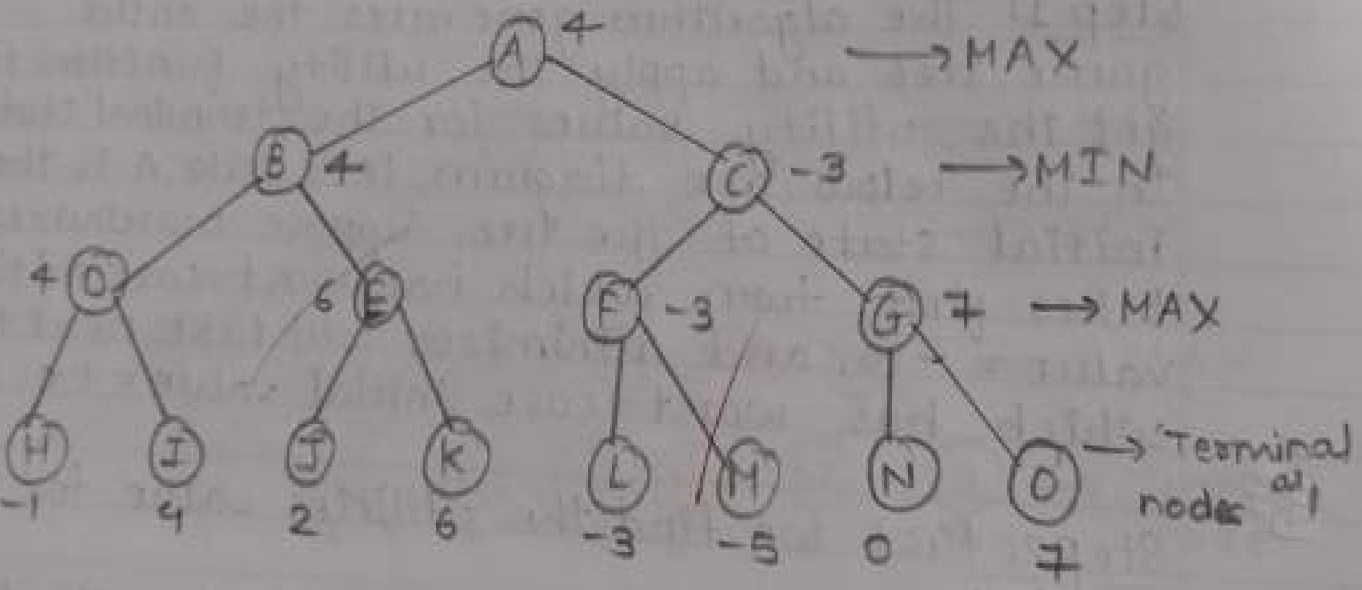
Step2: First we find the utilities value for the Maximizer.

- for node D - $\max(-1, -\infty) \Rightarrow \max(-1, 4) = 4$
- for node E - $\max(2, -\infty) \Rightarrow \max(2, 6) = 6$.
- for node F - $\max(-3, -\infty) \Rightarrow \max(-3, -5) = -3$
- for node G - $\max(0, -\infty) = \max(0, 7) = 7$.

step 3:


MAX
MIN
MAX
Terminal node.
A
B 4
C -3
4 D
6 E
F -3
G 7
H
I
J
K
L
M
N
O
-1
4
2
6
-3
-5
0
7

step 4:


A 4
MAX
B 4
C -3
MIN
4 D
6 E
F -3
G 7
MAX
H
I
J
K
L
M
N
O
Terminal node
-1
4
2
6
-3
-5
0
7

Step 3: In the next step, it's a turn for minimizer.

- 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 maximum of all nodes value.

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

CONCLUSION: The Min-Max algorithm always finds the optimal move for a player, assuming that the other player is also making optimal moves. This means that if the opponent also uses the Min-Max algorithm, the game will always end in a draw.

EXPERIMENT NO: 7

DATE:

AIM: Implement any one of the local search techniques.

THEORY:

- 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.

Different regions in the state space landscape:

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

- **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

- **Current State:** It is a state in a landscape diagram where an agent is currently present.

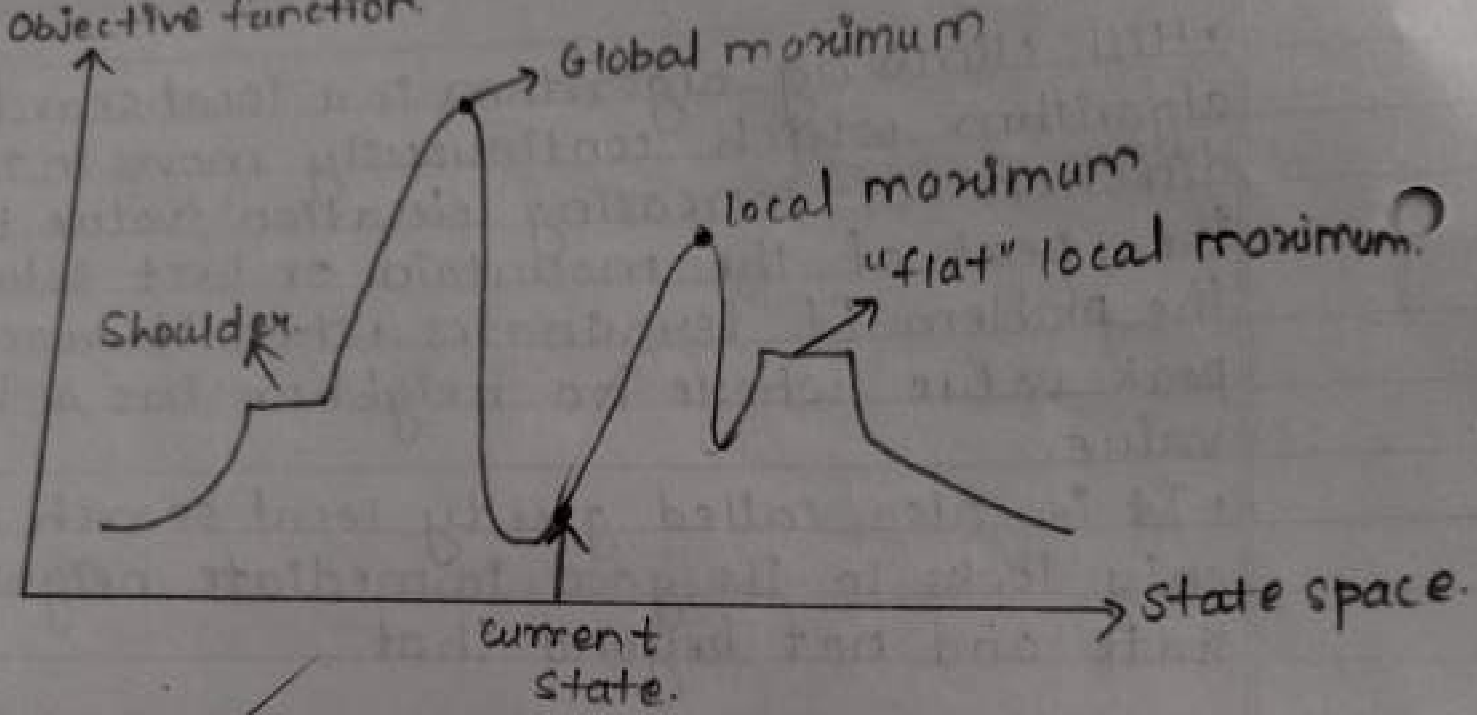


fig:- State-space diagram for Hill climbing.

• 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.

ALGORITHM:

Step1: Evaluate the initial state.
step 2: Loop until a solution is found
step 3: Select & apply an operator to the current state.
step 4: check new state, else if not better than the current state, then return to step 2.

CONCLUSION: It iteratively improves the current solution by making small change, and it's particularly suited for local optimization problems.

EXPERIMENT NO: 08.

Aim: Prove the goal sentence from the following set of statements in FOPL by applying forward, backward and resolution inference algorithms.

Theory:

- Inference engine: The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:

1. Forward chaining
2. Backward chaining

1. Forward chaining:

- Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine.
- Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

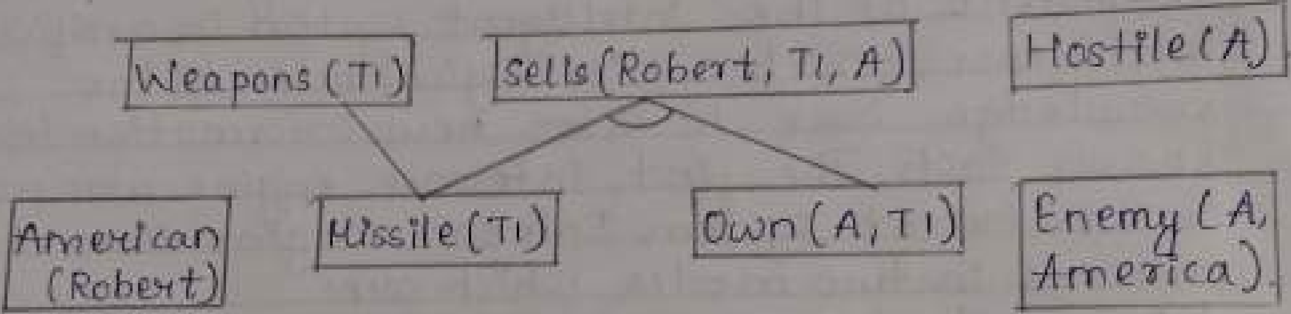
Example: "As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen." Prove that "Robert is criminal."

• Forward chaining proof:

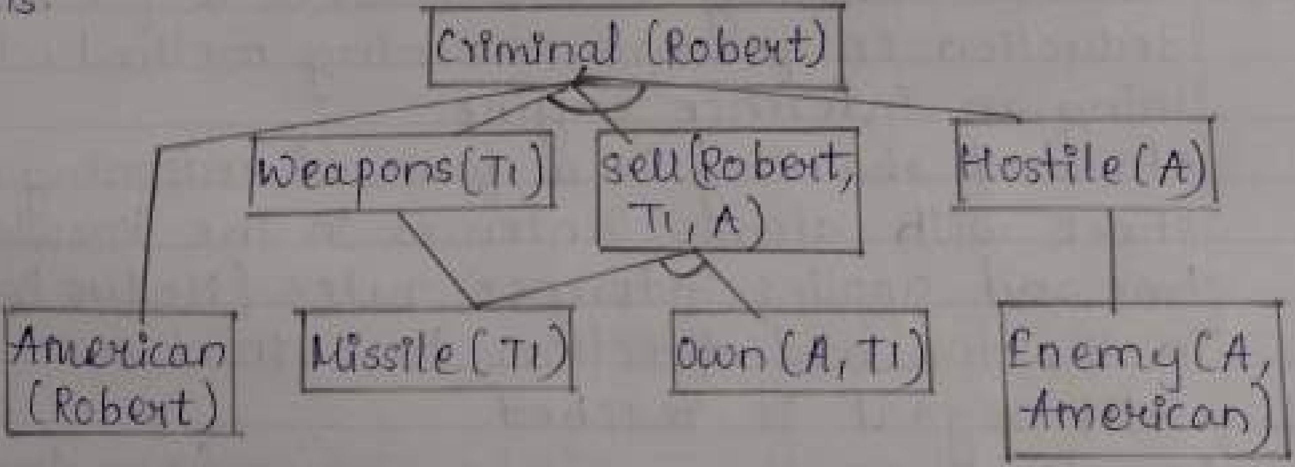
Step1:- In the first step we will start with the know facts and will choose the sentences which do not have implications.

American (Robert) | Missile (T1) | Owns (A, T1) | Enemy (A, America)

Step 2: At the second step, we will see those facts which infer from available facts and with satisfied premises



Step 3: At step-3, as we can check Rule-(1) is satisfied with the substitution {p/Robert, q/T1, r/A}, so we can add Criminal (Robert) which infers all the available facts



Hence, it is proved that Robert is Criminal using forward chaining approach.

Properties:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- forward-chaining approach is also called as data-driven as we reach to the goal using available data

2. Backward chaining:

- Backward-chaining is also known as a backward deduction or backward resoning method when using an inference engine.
- A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Example: "As per the law, it is a crime for an American to sell weapons to hostile nation. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen"

In backward-chaining, we will use the same above example

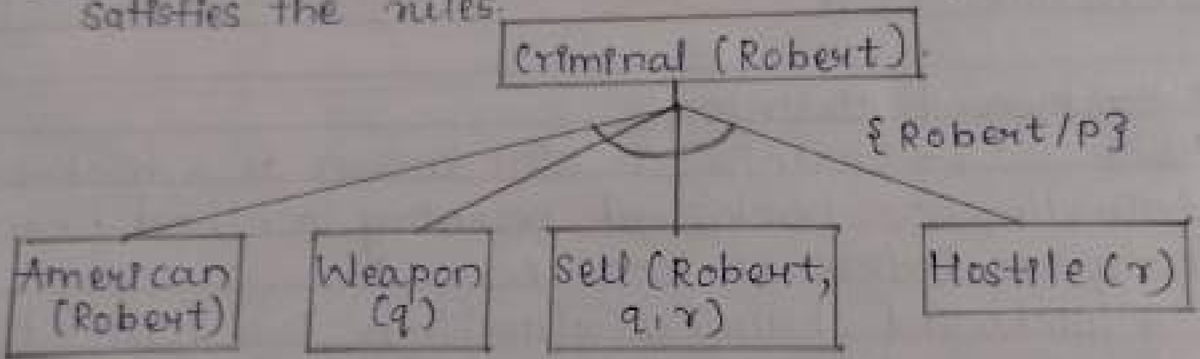
- **Backward-Chaining Proof:**

In Backword Chaining, we will start with our goal predicate, which is Criminal (Robert) and then infer further rules.

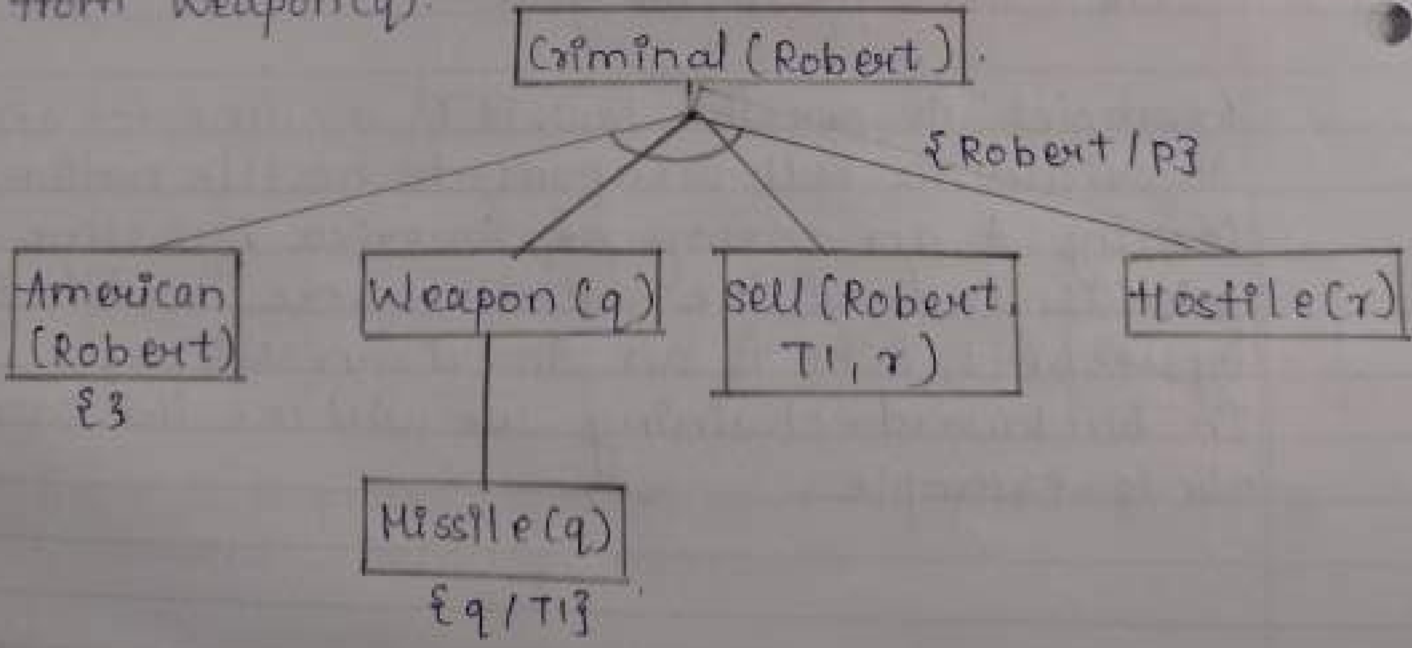
Step 1: At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. so our goal fact is "Robert is Criminal," so following is the predicate is

Criminal (Robert)

Step-2: We will infer other facts from goal fact which satisfies the rules.



Step-3: We will extract further fact Missile (q) which infer from weapon (q).



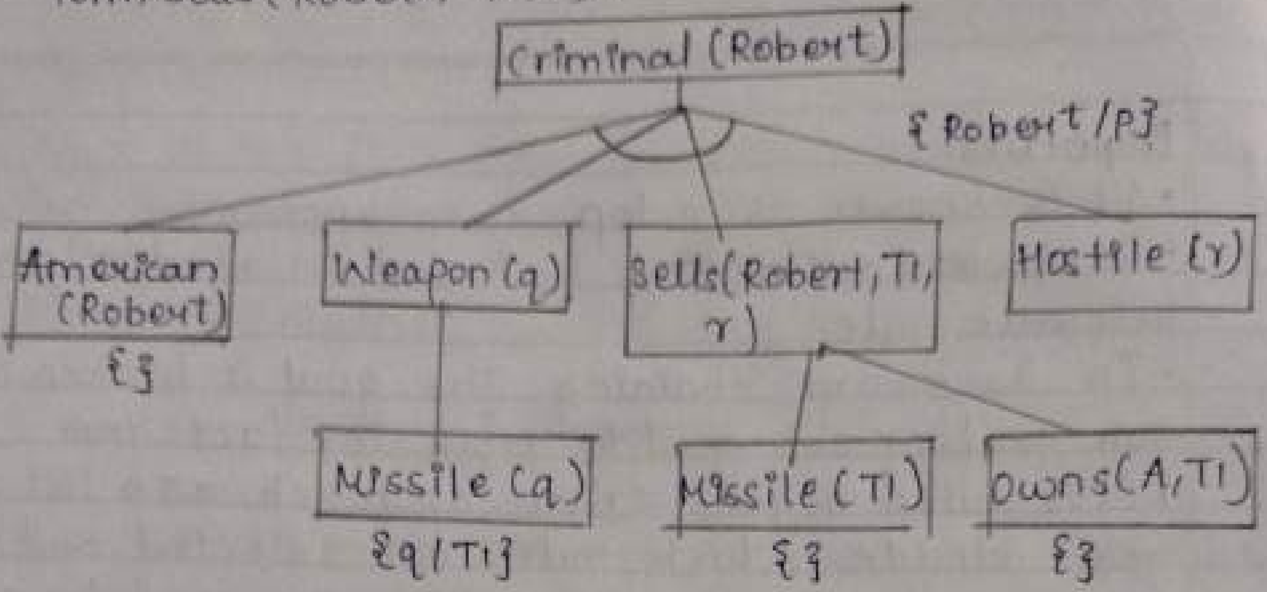
Properties:

- It is known as a top-down approach
- Backward - chaining is based on modus ponens inference rule
- In Backward chaining, the goal is broken into sub-goal or sub-goal to prove the facts true
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used
- Backward - chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications

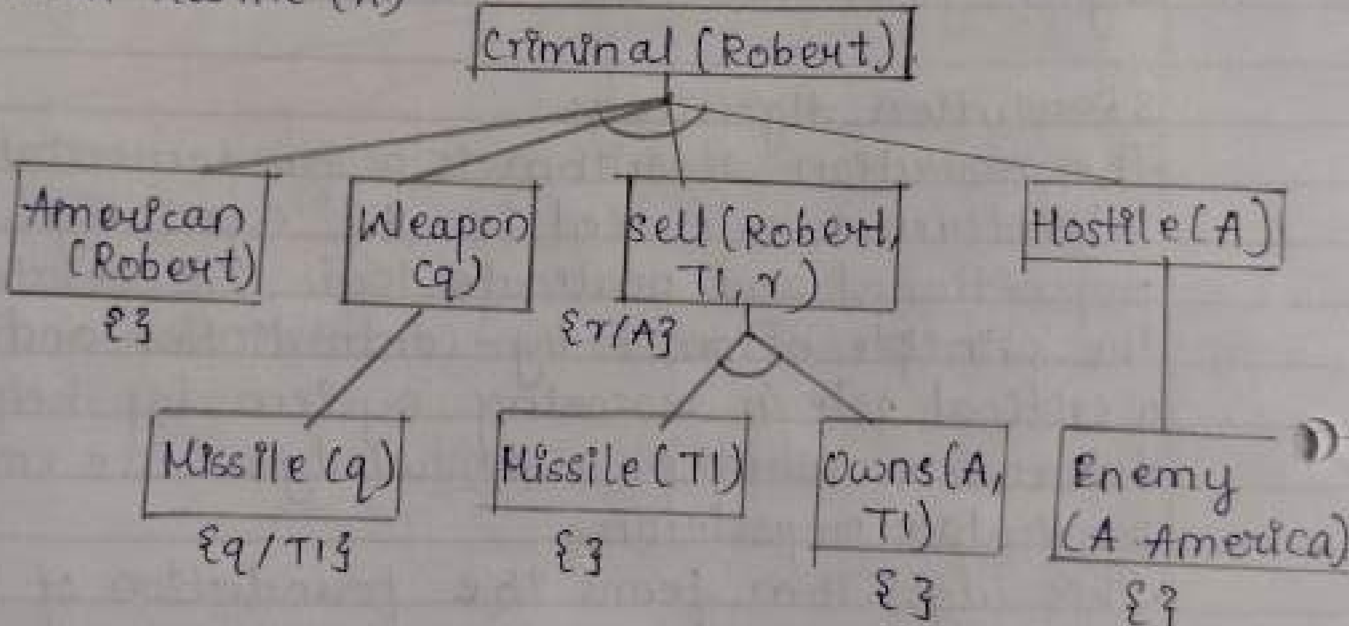
3 Resolution Algorithm:

- The resolution algorithm is a fundamental intereforence rule used in AI, especially within propositional and predicate logic. It operates on the principle of proof-by-contradiction and plays a critical role in reasoning system by helping determine whether a knowledge base entails a particular proposition.
- This algorithm forms the foundation of many AI systems, enabling them to reason logically and deduce new information from given facts.

step-4: we can infer facts Missile (T1) and owns(A, T1) form sells (Robert, T1, r).



Step-5: We can infer the fact Enemy (A, America) from Hostile (A)



Algorithm:

1. Input: The inputs to the algorithm are the knowledge base (KB) and the query (α).

2 Clause Conversion: The algorithm starts by converting KB $\wedge \sim\alpha$ into a set of clause in CNF

3 Loop & Resolution: In each iteration, the algo selects pairs of clauses and applies the resolution rule.

4. Termination: If no new clauses can be added. the algorithm returns false

Conclusion:

- We inferred the goal by starting from known facts and applying rules.
- We worked backwards from the goal, finding supporting facts.
- We negated the goal and derived a contradictio proving the goal true.

EXPERIMENT NO: 09

Aim: Create a Bayesian Network for the given Problem statement and draw inferences from it. ~~from~~

Theory:

- Bayesian Belief Network in artificial intelligence

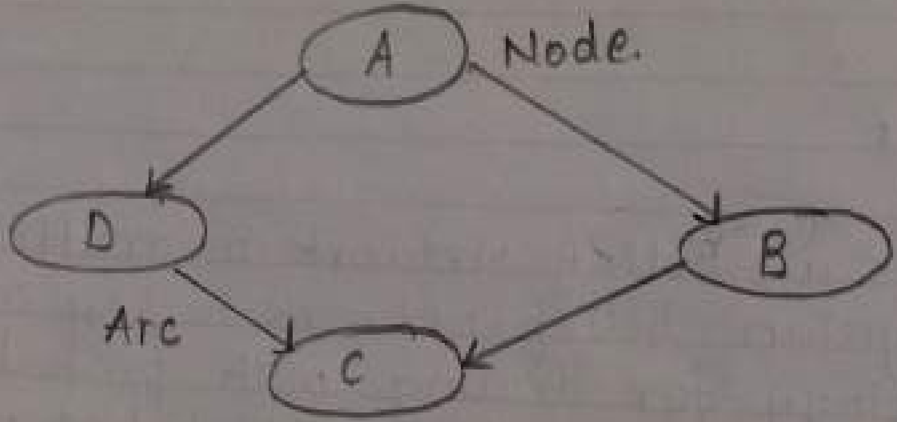
Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."
It is also called a Bayes network, belief network decision network, or Bayesian model

Bayesian networks are probabilistic, because these networks are built from a probablity distribution, and also use probability theory for prediction and anomaly detection.
It can be used for building models from data and experts opinions, and it consists of 2 parts

- Directed Acyclic Graph
- Table of conditional probabilities.

A Bayesian network graph is made up of nodes and Arcs (directed links),



- The generalized form of Bayesian network that represents and solve decision problem under certain knowledge is known as an Influence diagram.

- Each node corresponds to the random variables and a variable can be continuous or discrete
- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
- These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other.
- The Bayesian network has mainly two components:

1 Causal component
2 Actual numbers

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

Conclusion: Bayesian networks provide a powerful framework for understanding the probabilistics relationships between various variables in a complex system.

EXPERIMENT NO:10

Aim: Implement a Planning Agent

Theory:
A planning agent is a type of agent in computer science that is responsible for handling distributed tasks based on the AWSP-E algorithm. It respond to management request from the central agent and can execute multiple tasks in parallel.
The Planning Agent algorithm can be described as a Search-based approach to problem-solving. The core of this agent is a state-space search, where the agent explores possible actions and states until it finds a path to the goal.

Algorithm:

1. Initialize the frontier: start with a queue (or stack for DFS) containing the initial state and an empty sequence of actions.

2. Initialize the explored set: keep track of all states that have been visited to avoid re-exploration.

3. Loop until the frontier is empty:
a. Pop the current state and actions.

b. check if the current state is the goal:
c. Generate new states:
d. Add new states to the frontier
e. Mark the current state as explored.

4. Return failure: If the frontier becomes empty and no solution has been found, return that no solution exists.

- The algorithm is complete depends on the search strategy used.

1. BFS (Breadth-First Search):
It will eventually explore all possible states and find a solution if one exists, as it explores level by level.

2. DFS (Depth-First Search):
Not necessarily complete. DFS can get stuck in infinite loops in cyclic states spaces or explore infinite branches in some domains.

3. A*:
Complete if the heuristic is admissible (i.e., never overstimates the true cost to reach the goal)

Advantages:

1. Generalized Problem-Solving: They can be applied to a wide range of problems across various domains

2 Flexibility: Adaptable to different search algorithms (eg BFS, DFS, A*) depending on the problem.

Disadvantages:

1 High computation complexity in a large state spaces.

2. Memory intensive, especially with BFS and A*.

Conclusion: Planning Agents are effective for goal-oriented problem-solving but can be computationally and memory expensive, and may struggle with real-time or dynamic environments