Unit-II Knowledge Representation and Reasoning

Q.No	Part – A
1.	**Define game playing?** **Game Playing in AI** refers to the development of computer programs or agents that can play games intelligently against human players or other programs by making decisions based on the rules and objectives of the game. **Competitive environment:** Involves two or more players (agents) with conflicting goals. **Rules-based:** Each game has a defined set of rules (e.g., chess, tic-tac-toe). **Search and strategy:** Agents use **search algorithms** (like Minimax, Alpha-Beta Pruning) to explore possible moves. **Uncertainty and adversarial behavior:** The opponent's moves introduce uncertainty. Examples: **Chess-playing AI:** Deep Blue defeating Garry Kasparov. **Go-playing AI:** AlphaGo using deep learning and search. **Video game bots:** AI opponents in games like FIFA or Dota 2.
2.	**What is Mini –Max Strategy?** The **Mini–Max Strategy** is a decision-making algorithm used in **game-playing AI** to minimize the possible loss in a worst-case scenario. It is commonly used in **two-player, zero-sum games** (e.g., Chess, Tic-Tac-Toe), where one player's gain is another player's loss. **Key Idea** ● **MAX player:** Tries to **maximize** the score. ● **MIN player:** Tries to **minimize** the score. ● The algorithm assumes both players play optimally. ● It explores the **game tree** and uses **backtracking** to select the best move. **Working:** 1. Generate the **game tree** from the current state up to a certain depth. 2. Assign **utility values** (scores) to the leaf nodes: o Positive values for MAX (winning state). o Negative values for MIN (losing state). 3. **Propagate values upward**: o MAX node selects the **maximum value** of its children. o MIN node selects the **minimum value** of its children. 4. At the root, MAX chooses the move that gives the **best guaranteed outcome**. **Drawback** ● **Computationally expensive** because it explores the entire game tree. ● Improved by **Alpha-Beta pruning** to cut off unnecessary branches.
3.	**How is Knowledge represented?** Knowledge representation is the method of encoding information about the world into a form that a computer system can utilize to solve complex tasks like diagnosing a disease, conversing in natural language, or playing chess. **Types of Knowledge** 1. **Declarative Knowledge** – Facts about the world (e.g., “Paris is the capital of France”). 2. **Procedural Knowledge** – How to do something (e.g., steps to solve a puzzle). 3. **Meta-Knowledge** – Knowledge about knowledge (e.g., knowing which rule to apply). 4. **Heuristic Knowledge** – Rules of thumb (e.g., “If traffic is heavy, take an alternate route”). **Methods of Knowledge Representation** 1. **Logical Representation**

	○ Uses **propositional logic** or **predicate logic**. ○ Example: Human(x)→Mortal(x) (If x is human, then x is mortal) 2. **Semantic Network** ○ Graph-based structure with **nodes** (concepts) and **edges** (relationships). ○ Example: *"Dog is a Mammal"* → Node **Dog** connected to **Mammal** with "is-a" link. 3. **Frame Representation** ○ Data structure with **slots** and **fillers** to represent stereotypical situations. Example: Frame for **Car**: Slots = {Make, Model, Color, EngineType} 4. **Production Rules** ○ IF-THEN rules. Example: IF **temperature < 0°C** THEN **water is solid**. 5. **Ontology** ○ Structured representation of concepts and their relationships in a domain. ○ Used in **semantic web** and **knowledge graphs**. **Example** **Problem:** Represent "All humans are mortal. Socrates is a human." **Logical Representation:** ∀x(Human(x)→Mortal(x)), Human(Socrates)
4.	**Summarize propositional logic** Propositional Logic (PL) is a **formal system in AI** that deals with **propositions (statements)** that are either **true (T)** or **false (F)**. It uses **logical connectives** to build complex sentences from simple propositions. **Key Components** 1. **Propositions:** Atomic statements like P,Q,R Example: P: "It is raining." 2. **Logical Connectives:** ○ **¬P** → NOT P (Negation) ○ **P ∧ Q** → P AND Q (Conjunction) ○ **P ∨ Q** → P OR Q (Disjunction) ○ **P → Q** → If P then Q (Implication) ○ **P ↔ Q** → P if and only if Q (Biconditional) 3. **Truth Values:** Each proposition has **True (T)** or **False (F)**. **Syntax:** Rules to form valid sentences using propositions and connectives. Example: (P∧Q)→R **Semantics:** Assign meaning to sentences via **truth tables** **Example:** Statements: ● P: It is raining. ● Q: The ground is wet.

	$P \rightarrow Q$ (If it is raining, then the ground is wet.)
5.	**Explain in detail about resolution?** Resolution is an **inference rule** widely used in **propositional logic** and **first-order logic (FOL)** to derive conclusions or prove the unsatisfiability of a set of statements. It plays a key role in **automated theorem proving** and **logic-based AI systems**. Resolution is a **rule of inference** that works by **refutation**. The **principle**: If you want to prove a statement **α**, assume **¬α** and show that it leads to a **contradiction**. It works on **clauses in Conjunctive Normal Form (CNF)**. **Steps in Resolution Method** 1. **Convert all sentences into CNF (Conjunctive Normal Form)**: o A conjunction of disjunctions of literals. o Example: $(P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg P \vee \neg R)(P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg P \vee \neg R)$ 2. **Negate the statement to be proved** and add it to the knowledge base. 3. **Apply the resolution rule**: o From clauses **(A ∨ B)** and **(¬B ∨ C)**, we can infer **(A ∨ C)**. o This is called the **resolvent**. 4. **Repeat** until: o An **empty clause (⊥)** is derived → Contradiction → Statement is proved. o Or no new clauses can be derived → Statement is not provable.
6.	**What are the limitations of Propositional Logic?** **1. Lack of Expressiveness** ● PL can only represent **facts as true or false**. ● It **cannot represent relationships, properties, or quantifiers** like “for all” or “there exists”. ● Example: o In PL, you can say **P: "John is a human"**, but you **cannot express** “All humans are mortal” or “Some humans are doctors”. **2. No Support for Variables or Quantifiers** ● PL cannot use **variables**, **functions**, or **universal/existential quantifiers**. ● Example: o Statement: “Every student has a teacher.” o In PL, you would need a separate symbol for each student-teacher pair, which is inefficient. **3. No Inference about Objects or Classes** ● PL cannot **infer relationships** between objects or classes. ● For example: o From “All humans are mortal” and “Socrates is a human”, PL **cannot infer** “Socrates is mortal” because PL does not support such structured knowledge. **4. Knowledge Base Explosion** ● To represent a general rule for **n objects**, PL requires **n separate propositions**. ● Example: o For 100 students: ▪ P_1: Student1 is enrolled ▪ P_2: Student2 is enrolled ▪ … up to P_{100} o This becomes huge and unmanageable.

	5. No Support for Uncertainty or Degrees of Truth • PL works only with **True (T)** or **False (F)** values. • Real-world reasoning often requires **uncertainty handling**, which PL cannot represent. • Example: o "It will rain tomorrow with 70% probability" → Not possible in PL. **6. Cannot Represent Actions or Temporal Changes** • PL is **static**; it cannot represent how the world **changes over time**. • Example: o "The door is open now, but will be closed later" cannot be represented properly. **7. No Compact Representation of General Rules** • General statements like o "All birds can fly" must be written for **each bird individually** in PL.
7.	**Define pruning?** Pruning is the technique of **eliminating parts of the search space without completely exploring them**, to reduce computation time and memory usage, while still guaranteeing an optimal solution (if applicable). **Why Pruning is Needed?** • Large search trees (like in Chess or game playing) have **huge branching factors**. • Without pruning, algorithms like **Minimax** would evaluate all possible moves, which is computationally expensive. • Pruning improves **efficiency** without changing the final outcome. **Example** In **Alpha-Beta Pruning** (used with Minimax): • If we already have a move that guarantees a win, we **stop exploring other worse moves**. • For example: o If MAX has found a move with value 8, and another branch cannot possibly exceed 8, we **prune** that branch.
8.	**Explain the PEAS description of Wumpus World problem?** The **PEAS** framework stands for **Performance measure, Environment, Actuators, and Sensors**. It is used to describe an intelligent agent's task environment in AI. Let's apply it to the **Wumpus World** problem. **What is Wumpus World?** • A cave consisting of **rooms connected by passages**. • The agent must navigate the cave to find **gold** and avoid **pits** and the **Wumpus** (a monster). • The agent can perceive stench (near Wumpus), breeze (near pits), glitter (gold), bump (into wall), and scream (when Wumpus dies). **1. Performance Measure (P)** How do we evaluate success? • **+1000** for finding gold. • **-1000** for getting killed by Wumpus or falling into a pit. • **-1** for each action (to encourage efficiency). • **-10** for using the arrow. • **Objective:** Maximize score by getting gold and exiting safely with minimal actions. **2. Environment (E)** What is the agent's world like?

	• A **grid of squares** (e.g., 4×4 or larger).
	• Each square may contain:
	  o **Wumpus** (monster)
	  o **Pit**
	  o **Gold**
	  o Or be empty
	• Properties:
	  o **Partially observable** (agent only senses local indicators).
	  o **Discrete** (finite grid squares).
	  o **Static** (world doesn't change except Wumpus dying).
	  o **Deterministic** (actions have predictable results).
	  o **Sequential** (previous actions affect future outcomes).
	3. Actuators (A)
	What actions can the agent take?
	• **Move Forward**
	• **Turn Left**
	• **Turn Right**
	• **Grab** (pick up gold)
	• **Shoot** (use arrow to kill Wumpus)
	• **Climb** (exit the cave)
	4. Sensors (S)
	What can the agent perceive?
	• **Stench** → Wumpus is in an adjacent square.
	• **Breeze** → Pit is in an adjacent square.
	• **Glitter** → Gold is in the current square.
	• **Bump** → Agent hit a wall.
	• **Scream** → Wumpus has been killed

9.	**List out the basic elements of First-Order logic?**
	The **basic elements of First-Order Logic (FOL)** are the building blocks used to represent knowledge in a structured way. They include:
	1. Constants
	• Represent **specific objects** in the domain.
	• Example: John, Paris, 3.
	2. Variables
	• Represent **unspecified or generic objects**.
	• Example: x, y, z.
	3. Predicates
	• Represent **relations** among objects or **properties** of objects.
	• Example:
	  o Loves(John, Mary) → John loves Mary.
	  o IsHuman(x) → x is human.
	4. Functions
	• Map objects to other objects in the domain.
	• Example:
	  o Mother(x) → returns the mother of x.
	  o Add(2,3) → 5.
	5. Connectives (Logical Operators)
	• Combine or modify statements.
	• **Types:**
	  o ¬ (Negation)

- ∧ (Conjunction / AND)
- ∨ (Disjunction / OR)
- → (Implication)
- ↔ (Biconditional)

6. Quantifiers

- Express **universality** or **existence**.
- **Types:**
  - **Universal (∀)**: "For all"
    - Example: ∀x IsHuman(x) → Mortal(x) → All humans are mortal.
  - **Existential (∃)**: "There exists"
    - Example: ∃x Loves(x, Mary) → There exists someone who loves Mary.

7. Equality

- Used to state that two terms refer to the same object.
- Example: Father(John) = Peter.

8. Sentences / Well-formed Formulas

- Complete statements built from the above elements that can be true or false.
- Example:
  - ∀x (IsHuman(x) → Mortal(x))

10.

Name the two types of quantifiers and explain with an example?

The **two types of quantifiers in First-Order Logic (FOL)** are:

1. Universal Quantifier (∀)

- **Meaning:** "For all" or "For every."
- **Symbol:** ∀
- **Purpose:** States that a property or relation holds for **all elements** in the domain.
- **Example:**
  ∀x (Human(x) → Mortal(x))
  Interpretation: For every x, if x is a human, then x is mortal.
  (All humans are mortal.)

2. Existential Quantifier (∃)

- **Meaning:** "There exists" or "For some."
- **Symbol:** ∃
- **Purpose:** States that there is **at least one element** in the domain for which the property or relation holds.
- **Example:**
  ∃x (Human(x) ∧ Scientist(x))
  Interpretation: There exists some x such that x is a human and a scientist.
  (There is at least one human who is a scientist.)

11.

Compare Forward and Backward chaining.

S.No.	Forward Chaining	Backward Chaining
1.	Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the required facts that support the goal.
2.	It is a bottom-up approach	It is a top-down approach
3.	Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.

	4.	Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
	5.	Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
	6.	Forward chaining is suitable for the planning, monitoring, control, and interpretation application.	Backward chaining is suitable for diagnostic, prescription, and debugging application.
	7.	Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number o possible conclusions.
	8.	It operates in the forward direction.	It operates in the backward direction.
	9.	Forward chaining is aimed for any conclusion.	Backward chaining is only aimed for the required data.

12.

Discuss in detail about resolution inference rule with its equation.

Resolution is a **rule of inference** used in **Propositional Logic** and **First-Order Logic** for **automated theorem proving**. It works by **refuting** the negation of a statement using **contradiction**.

The **resolution inference rule** allows us to infer a new clause by eliminating complementary literals from two clauses.

- A **literal** is either an atomic proposition (e.g., P) or its negation (e.g., ¬P).
- Two literals are **complementary** if one is the negation of the other (e.g., P and ¬P).

If we have two clauses:

$(C1 \vee L), (C2 \vee \neg L)$

Then we can infer:

$(C1 \vee C2)$

Here:

- C_1 and C_2 are (possibly empty) disjunctions of literals.
- L and ¬L are complementary literals.
- The new clause $(C_1 \vee C_2)$ is called the **resolvent**.

Steps in Resolution (Propositional Logic)

1. **Convert** all sentences in the knowledge base (KB) and the negation of the query into **Conjunctive Normal Form (CNF)**.
2. **Break** them into **clauses** (disjunctions of literals).
3. **Apply resolution rule** repeatedly on pairs of clauses that contain complementary literals.
4. If we derive an **empty clause (⊥)**, then the KB **entails the query**.

13.

Give an outline of neuro-fuzzy inference. How does it combine neural networks and fuzzy logic?

A **Neuro-Fuzzy Inference System (NFIS)** is a hybrid system that combines the **learning ability of neural networks** with the **reasoning capability of fuzzy logic**. It is widely used for control systems, decision-making, and modeling complex, nonlinear systems.

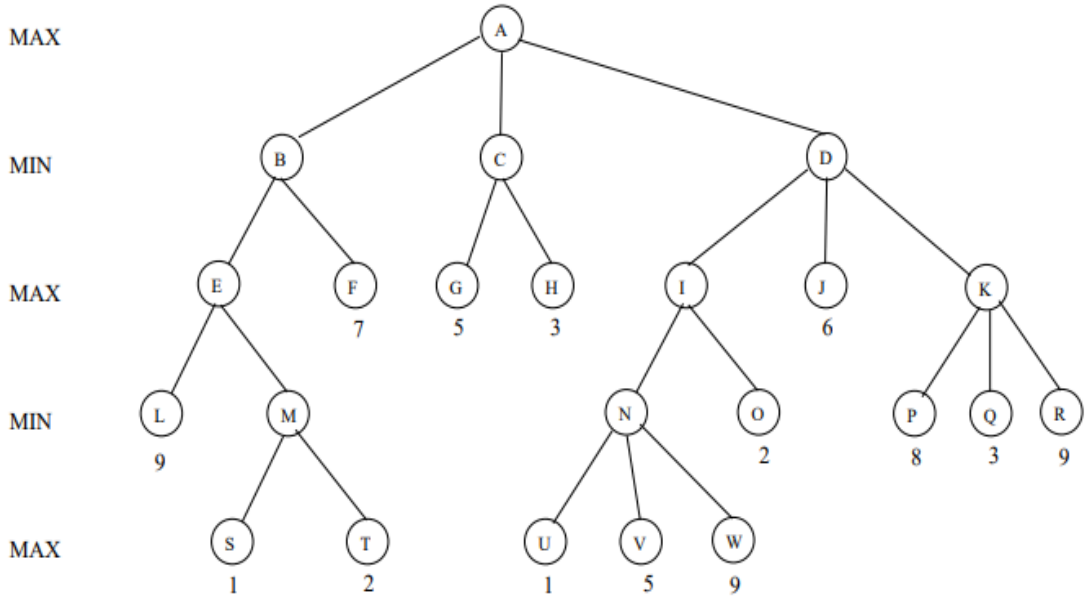
- **Fuzzy Logic**: Handles uncertainty, imprecision, and linguistic rules (e.g., "If temperature is high, then fan speed is fast").

	● **Neural Networks**: Learn from data, adjust parameters automatically, and model complex nonlinear relationships. **Neuro-Fuzzy System**: Uses **fuzzy rules and sets**, but the **parameters of membership functions and rules are tuned using neural network learning algorithms** (e.g., backpropagation). Most commonly implemented as **ANFIS (Adaptive Neuro-Fuzzy Inference System)**. Typical layers: 1. **Layer 1 – Input Layer** o Inputs are crisp values (e.g., temperature, speed). o Each node corresponds to an input variable. 2. **Layer 2 – Fuzzification Layer** o Converts crisp inputs into fuzzy values using **membership functions (MFs)**. o Example: "Low," "Medium," "High" represented by Gaussian or triangular MFs. o **Adjustable parameters** of MFs are tuned by learning. 3. **Layer 3 – Rule Layer** o Each node represents a fuzzy rule (e.g., IF x is A AND y is B THEN z is C). o The output of this layer is the **firing strength** of the rule. 4. **Layer 4 – Normalization Layer** o Normalizes firing strengths of all rules. 5. **Layer 5 – Defuzzification / Output Layer** o Computes the final crisp output using methods like **weighted average**. **Inference Mechanism** ● Uses **fuzzy inference** (commonly Sugeno or Mamdani model). ● Neural network learning adjusts: o Membership function shapes. o Rule weights. o Output parameters. **Combines Neural Networks and Fuzzy Logic** ● **Neural Network Contribution**: o Provides **learning capability** (e.g., gradient descent, backpropagation). o Automatically **tunes membership functions** and rule parameters. o Handles **adaptation** with new data. ● **Fuzzy Logic Contribution**: o Provides **interpretability** (rules are human-readable). o Handles **imprecision** and **uncertainty**. o Represents knowledge in linguistic terms.
14.	**Interpret about neural network in AI? How does it mimic the human brain?** A **neural network** in Artificial Intelligence (AI) is a **computational model inspired by the structure and functioning of the human brain**. It is used for **pattern recognition, prediction, classification, and decision-making**. Neural networks form the foundation of **deep learning**. **Neural Network** ● A **neural network** consists of **interconnected nodes (neurons)** organized in layers: o **Input layer**: Takes the input features. o **Hidden layers**: Perform computations through weighted connections. o **Output layer**: Produces the final result (classification, prediction, etc.). ● Each connection has a **weight** that determines the strength of the signal. ● Uses **activation functions** (e.g., sigmoid, ReLU) to introduce non-linearity. **How It Mimics the Human Brain**

	<table><tr><th>Human Brain</th><th>Artificial Neural Network</th></tr><tr><td>Brain has billions of neurons</td><td>ANN has nodes (artificial neurons)</td></tr><tr><td>Neurons connected by synapses</td><td>Nodes connected by weighted links</td></tr><tr><td>Neurons activate when signal exceeds threshold</td><td>Neurons activate based on activation function</td></tr><tr><td>Brain learns by adjusting synaptic strength</td><td>ANN learns by adjusting weights via training</td></tr><tr><td>Uses parallel processing</td><td>ANN also works in parallel</td></tr></table> **Similarities with the Brain** • Both process information through **networks of simple units**. • Both can **learn from examples** (experience). • Both exhibit **distributed and parallel computation**. **Differences** • **Scale**: Human brain has ~86 billion neurons; ANN has thousands to millions. • **Energy efficiency**: Brain is more efficient and consumes less power. • **Flexibility**: Brain is better at generalization and multitasking.
15.	**Write the semantics of Bayesian network?** A **Bayesian Network (BN)** is a **probabilistic graphical model** that represents a set of random variables and their **conditional dependencies** using a **directed acyclic graph (DAG)**. The **semantics** of a Bayesian Network defines how the **graph structure** and the **conditional probability tables (CPTs)** specify a **joint probability distribution (JPD)** over all variables. **Key Components** • **Nodes**: Represent random variables X1,X2,...,Xn • **Edges**: Directed edges show **causal or dependency relationships**. • **Parents**: For a node Xi, its parents are the nodes with edges pointing to Xi • **Conditional Probability Distribution**: Each node has a **CPT** specifying P(Xi | Parents(Xi) **Semantics** The semantics are based on the **chain rule of probability** and the **conditional independence assumptions** encoded by the network. For a Bayesian Network with variables X1,X2,...,Xn, the **joint probability distribution (JPD)** is given by: $$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i)).$$ This means: • The probability of the whole system is the product of the **local conditional probabilities**. • **Each variable is conditionally independent of its non-descendants given its parents**.

Q.No	Part - B

1.



Interpret the solution for the above problem using Min-Max Algorithm

Answer:

Min-Max algorithm is a decision-making algorithm used in artificial intelligence, particularly in game theory and computer games. It is designed to minimize the possible loss in a worst-case scenario (hence "min") and maximize the potential gain (therefore "max").

Working of Min-Max Process in AI

Min-Max algorithm involves two players: the maximizer and the minimizer, each aiming to optimize their own outcomes.

Players Involved

Maximizing Player (Max):

- Aims to maximize their score or utility value.
- Chooses the move that leads to the highest possible utility value, assuming the opponent will play optimally.

Minimizing Player (Min):

- Aims to minimize the maximizer's score or utility value.
- Selects the move that results in the lowest possible utility value for the maximizer, assuming the opponent will play optimally.

The interplay between these two players is central to the Min-Max algorithm, as each player attempts to outthink and counter the other's strategies.

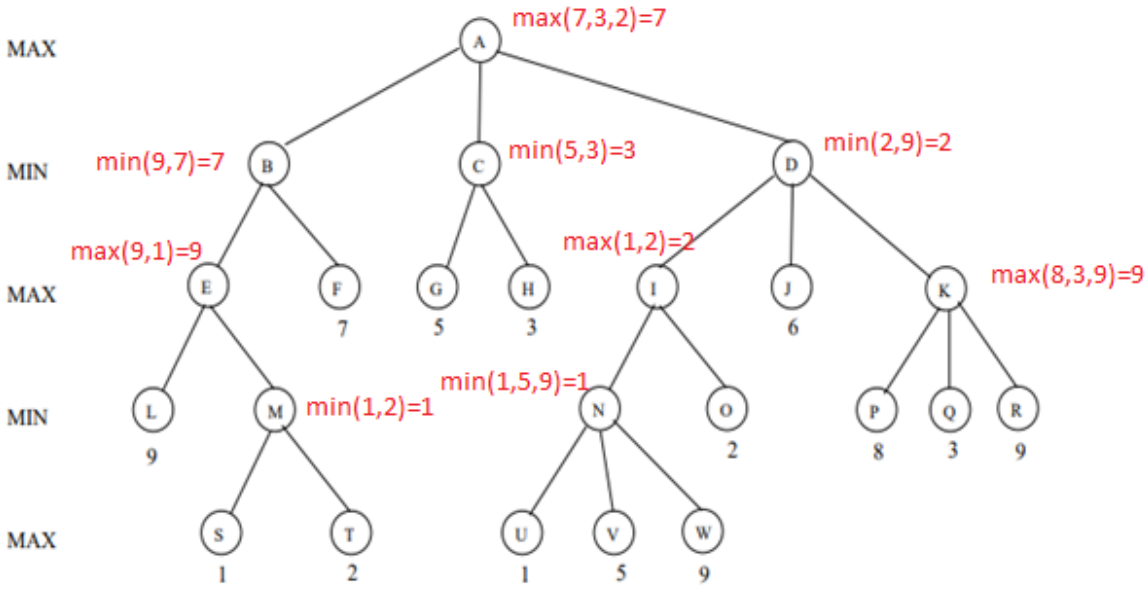
Pseudocode for Min-Max Algorithm

```
def minmax(state, depth, maximizing_player):
    if is_terminal(state) or depth == 0:
        return utility(state)

    if maximizing_player:
        max_eval = -infinity
        for action in actions(state):
            eval = minmax(result(state, action), depth - 1, False)
            max_eval = max(max_eval, eval)
        return max_eval
```

```
    else:
        min_eval = infinity
        for action in actions(state):
            eval = minmax(result(state, action), depth - 1, True)
            min_eval = min(min_eval, eval)
        return min_eval
```

Solution for the above problem using Min-Max Algorithm

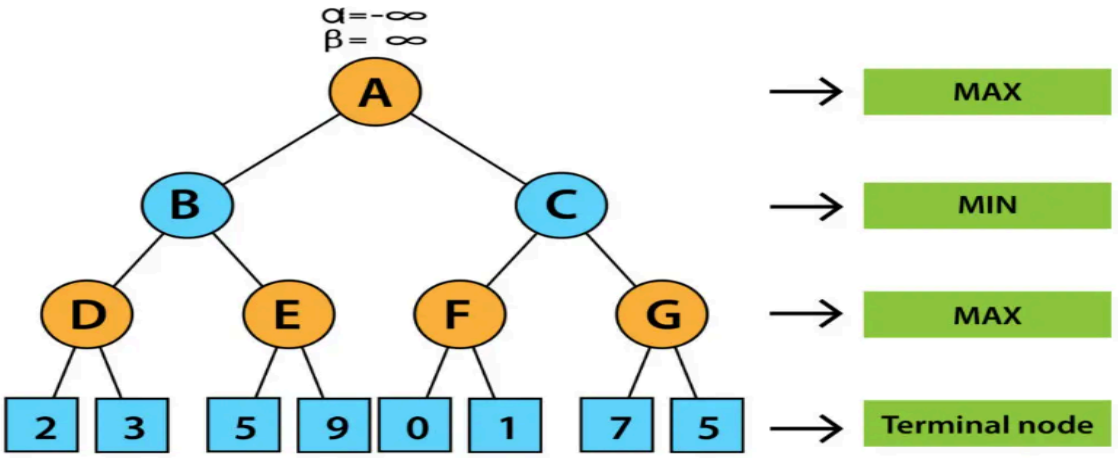


Optimal play

- **A → B** (since 7 is best for MAX),
- Opponent at **B (MIN)** chooses **F** (value 7) over **E (9)**,
- Outcome: **utility 7**.

So, by Min–Max, the optimal move from the root is to go to **B**, and with optimal counterplay the game value is **7** (path A → B → F).

2.

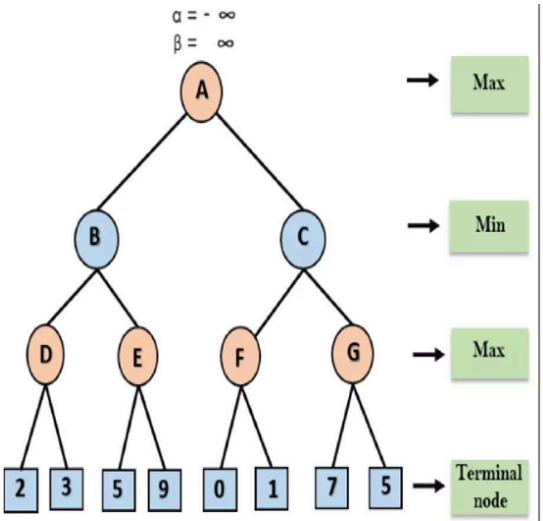


Interpret the solution for the above problem using α-β pruning Algorithm

Answer:

Step 1:

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

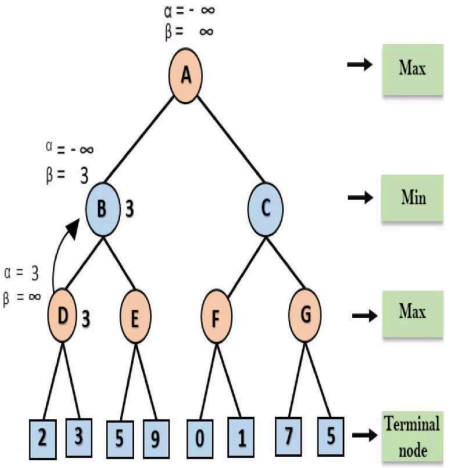


Step 2:

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

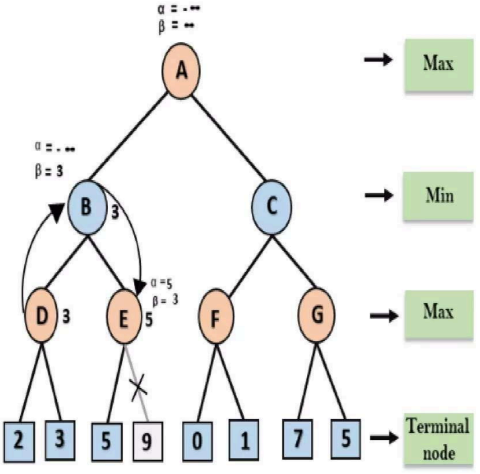
Step 3:

- Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.
- In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.



Step 4:

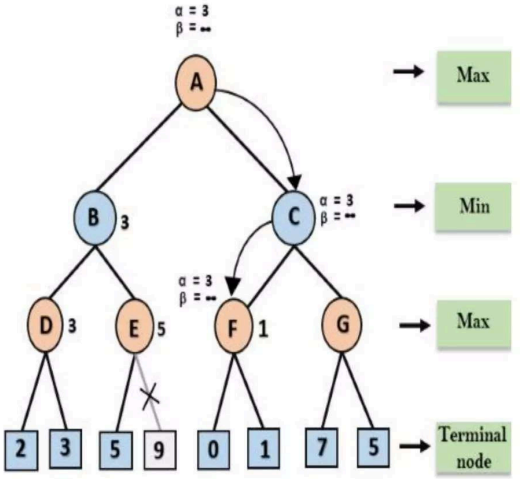
- At node E, Max will take its turn, and the value of alpha will change.
- The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be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



Step 5:

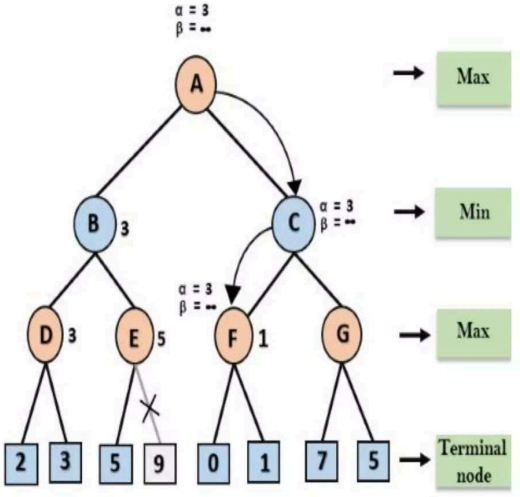
- At next step, algorithm again backtrack the tree, from node B to node A.
- At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

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



Step 6:

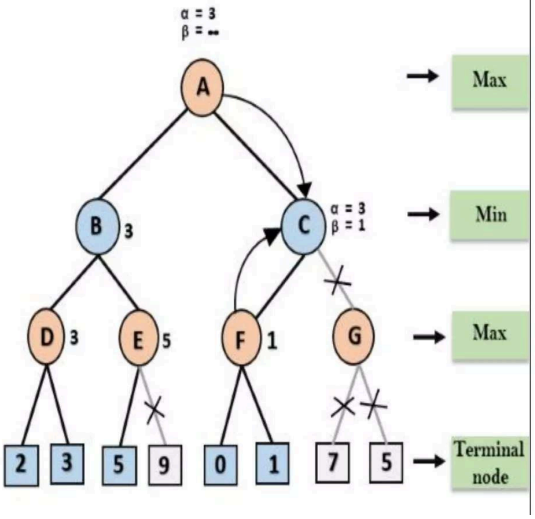
- At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



Step 7:

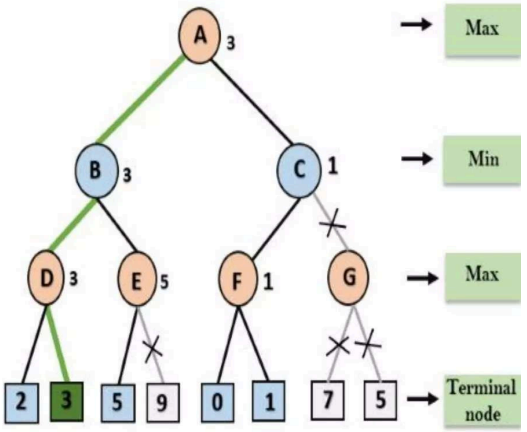
- Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1.

- Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8:

- **Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3.**
- **Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed.**
- **Hence the optimal value for the maximizer is 3 for this example.**



3. **Interpret the types of inference rules with appropriate examples along with its truth table.**

Modus Ponens:

The Modus Ponens rule is one of the most important rules of inference, and it states that if P and P → Q is true, then we can infer that Q will be true. It can be represented as:

Notation for Modus ponens: $\frac{P \rightarrow Q, \quad P}{\therefore Q}$

Example:

Statement-1: "If I am sleepy then I go to bed" ==> P→ Q

Statement-2: "I am sleepy" ==> P Conclusion: "I go to bed." ==> Q.

Hence, we can say that, if P→ Q is true and P is true then Q will be true.

Q	$P \rightarrow Q$
0	0
1	1
0	0
1	1

Proof by Truth table:

Modus Tollens:

The Modus Tollens rule state that if P→ Q is true and ¬ Q is true, then ¬ P will also true. It can be represented as:

Notation for Modus Tollens: $\frac{P \rightarrow Q, \quad \sim Q}{\sim P}$

Statement-1: "If I am sleepy then I go to bed" ==> P→ Q Statement-2: "I do not go to the bed."==> ~Q Statement-3: Which infers that "I am not sleepy" => ~P

P	Q	$\sim P$	$\sim Q$	$P \rightarrow Q$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

Hypothetical Syllogism:

The Hypothetical Syllogism rule state that if P⟶R is true whenever P⟶Q is true, and Q⟶R is true. It can be represented as the following notation:

Example:

Statement-1: If you have my home key then you can unlock my home. P⟶Q

Statement-2: If you can unlock my home then you can take my money. Q⟶R

Conclusion: If you have my home key then you can take my money. P⟶R

Proof by truth table:

Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$P \rightarrow R$
0	0	1	1	1
0	1	1	1	1
1	0	1	0	1
1	1	1	1	1
0	0	0	1	1
0	1	0	1	1
1	0	1	0	0
1	1	1	1	1

Disjunctive Syllogism:

The Disjunctive syllogism rule state that if PVQ is true, and ¬P is true, then Q will be true. It can be represented as:

Notation of Disjunctive syllogism: $\frac{P \vee Q, \ \neg P}{Q}$

Example:

Statement-1: Today is Sunday or Monday. ==>PVQ Statement-2: Today is not Sunday. ==> ¬P Conclusion: Today is Monday. ==> Q

Proof by truth-table:

P	Q	$\neg P$	$P \vee Q$
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1

Addition:

The Addition rule is one the common inference rule, and it states that If P is true, then PVQ will be true.

Notation of Addition: $\frac{P}{P \vee Q}$

Example:

Statement: I have a vanilla ice-cream. ==> P

Statement-2: I have Chocolate ice-cream.

Conclusion: I have vanilla or chocolate ice-cream. ==> (P∨Q)

Proof by Truth-Table:

	Q	$P \vee Q$	
	0	0	
	0	1	←
	1	1	
	1	1	←

Simplification:

The simplification rule state that if P∧ Q is true, then Q or P will also be true. It can be represented as:

Notation of Simplification rule: $\frac{P \wedge Q}{Q}$ Or $\frac{P \wedge Q}{P}$

Proof by Truth-Table:

P	Q	$P \wedge Q$	
0	0	0	
1	0	0	
0	1	0	
1	1	1	←

Resolution:

The Resolution rule state that if P∨Q and ¬ P∧R is true, then Q∨R will also be true. It can be represented as

Notation of Resolution $\frac{P \vee Q,\ \neg P \wedge R}{Q \vee R}$

Proof by Truth-Table:

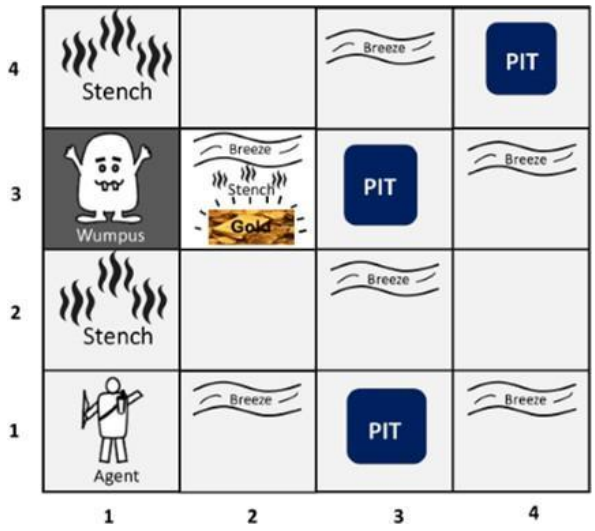
	Q	R	$P \vee Q$	¬P∧R	$Q \vee R$	
	0	0	0	0	0	
	0	1	0	0	1	
	1	0	1	1	1	←
	1	1	1	1	1	←
	0	0	1	0	0	
	0	1	1	0	1	
	1	0	1	0	1	
	1	1	1	0	1	←

4.

Explain in detail about the Wumpus world problem and prove it using Knowledge Representation?

The Wumpus world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So the agent goal is to find the gold and climb out the cave

without fallen into Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit.
Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.



There are also some components which can help the agent to navigate the cave. These components are given as follows:
The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
There will be glitter in the room if and only if the room has gold.
The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.

PEAS description of Wumpus world:
To explain the Wumpus world we have given PEAS description as below:
Performance measure:
o +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:

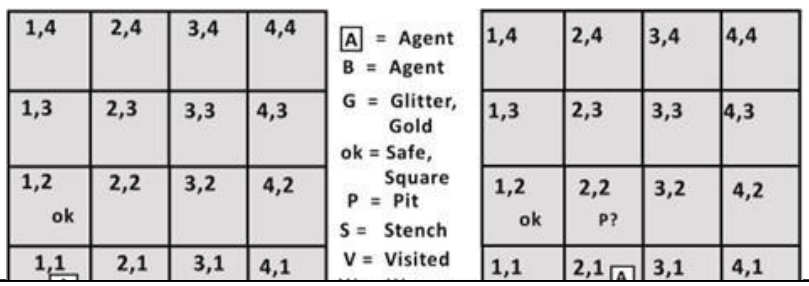
Partially observable: The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
Deterministic: It is deterministic, as the result and outcome of the world are already known.
Sequential: The order is important, so it is sequential.
Static: It is static as Wumpus and Pits are not moving.
Discrete: The environment is discrete.
One agent: The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

Exploring the Wumpus world:

Now we will explore the Wumpus world and will determine how the agent will find its goal by applying logical reasoning.

Agent's First step:

Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so to represent on the below diagram (a) that room is safe we will add symbol OK. Symbol A is used to represent agent, symbol B for the breeze, G for Glitter or gold, V for the visited room, P for pits, W for Wumpus.
At Room [1,1] agent does not feel any breeze or any Stench which means the adjacent squares are also OK.



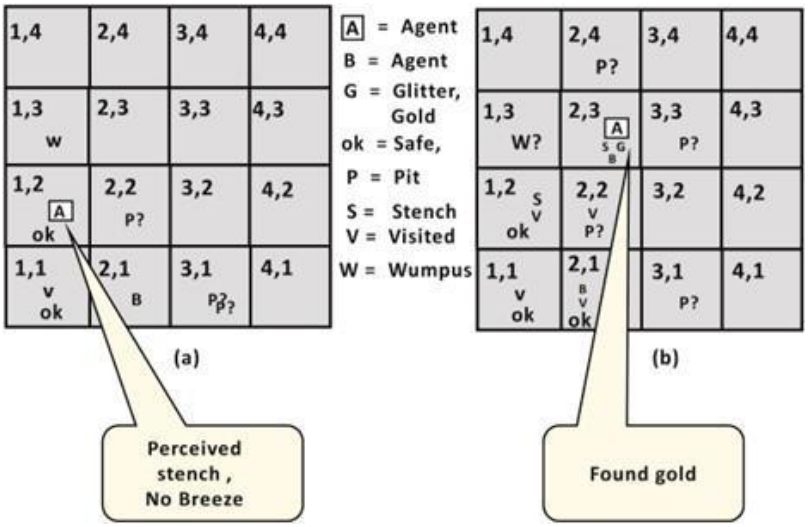
Agent's second Step:

Now agent needs to move forward, so it will either move to [1, 2], or [2,1]. Let's suppose agent moves to the room [2, 1], at this room agent perceives some breeze which means Pit is around this room. The pit can be in [3, 1], or [2,2], so we will add symbol P? to say that, is this Pit room?

Now agent will stop and think and will not make any harmful move. The agent will go back to the [1, 1] room. The room [1,1], and [2,1] are visited by the agent, so we will use symbol V to represent the visited squares.

Agent's third step:

At the third step, now agent will move to the room [1,2] which is OK. In the room [1,2] agent perceives a stench which means there must be a Wumpus nearby. But Wumpus cannot be in the room [1,1] as by rules of the game, and also not in [2,2] (Agent had not detected any stench when he was at [2,1]). Therefore agent infers that Wumpus is in the room [1,3], and in current state, there is no breeze which means in [2,2] there is no Pit and no Wumpus. So it is safe, and we will mark it OK, and the agent moves further in [2,2].



Agent's fourth step:

At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3]. At room [2,3] agent perceives glitter, so it should grab the gold and climb out of the cave.

Knowledge-base for Wumpus world

As in the previous topic we have learned about the wumpus world and how a knowledge-based agent evolves the world. Now in this topic, we will create a knowledge base for the wumpus world, and will derive some proves for the Wumpus-world using propositional logic.

The agent starts visiting from first square [1, 1], and we already know that this room is safe for the agent. To build a knowledge base for wumpus world, we will use some

rules and atomic propositions. We need symbol [i, j] for each location in the wumpus world, where i is for the location of rows, and j for column location.

1,4	2,4 P?	3,4	4,4
1,3 W?	2,3 S G B	3,3	4,3
1,2	2,2 V P?	3,2	4,2
1,1 A ok	2,1 B V ok	3,1 P?	4,1

Atomic proposition variable for Wumpus world:

Let Pi,j be true if there is a Pit in the room [i, j].

Let Bi,j be true if agent perceives breeze in [i, j], (dead or alive).

Let Wi,j be true if there is wumpus in the square[i, j].

Let Si,j be true if agent perceives stench in the square [i, j].

Let Vi,j be true if that square[i, j] is visited.

Let Gi,j be true if there is gold (and glitter) in the square [i, j].

Let OKi,j be true if the room is safe.

*Note: For a 4 * 4 square board, there will be 7*4*4= 122 propositional variables.*

Some Propositional Rules for the wumpus world:

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

Note: lack of variables gives us similar rules for each cell.

Representation of Knowledgebase for Wumpus world:

Following is the Simple KB for wumpus world when an agent moves from room [1, 1], to room [2,1]:

1	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	V_{11}	OK_{11}
	$\neg P_{12}$	-----	----	$\neg V_{12}$	OK_{12}
1	$\neg P_{21}$	B_{21}	$\neg G_{21}$	V_{21}	OK_{21}

Here in the first row, we have mentioned propositional variables for room[1,1], which is showing that room does not have wumpus(¬ W11), no stench (¬S11), no Pit(¬P11), no breeze(¬B11), no gold (¬G11), visited (V11), and the room is Safe(OK11).
In the second row, we have mentioned propositional variables for room [1,2], which is showing that there is no wumpus, stench and breeze are unknown as an agent has not visited room [1,2], no Pit, not visited yet, and the room is safe.
In the third row we have mentioned propositional variable for room[2,1], which is showing that there is no wumpus(¬ W21), no stench (¬S21), no Pit (¬P21), Perceives breeze(B21), no glitter(¬G21), visited (V21), and room is safe (OK21).

Prove that Wumpus is in the room (1, 3)

We can prove that wumpus is in the room (1, 3) using propositional rules which we have derived for the wumpus world and using inference rule.

Apply Modus Ponens with ¬S11 and R1:

We will firstly apply MP rule with R1 which is ¬S11 → ¬ W11 ^ ¬ W12 ^ ¬ W21, and ¬S11 which will give the output ¬ W11 ^ W12 ^ W12.

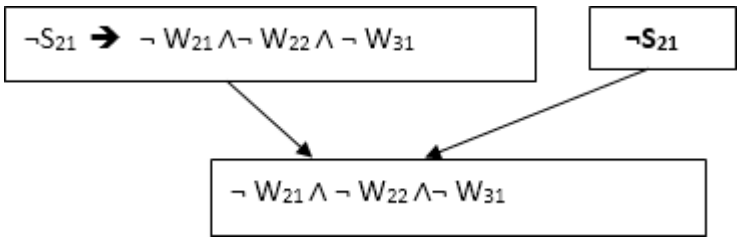
Apply And-Elimination Rule:

After applying And-elimination rule to ¬ W11 ∧ ¬ W12 ∧ ¬ W21, we will get three statements:

¬ W11, ¬ W12, and ¬W21.

Apply Modus Ponens to ¬S21, and R2:

Now we will apply Modus Ponens to ¬S21 and R2 which is ¬S21 → ¬ W21 ∧¬ W22 ∧ ¬ W31, which will give the Output as ¬ W21 ∧ ¬ W22 ∧¬ W31



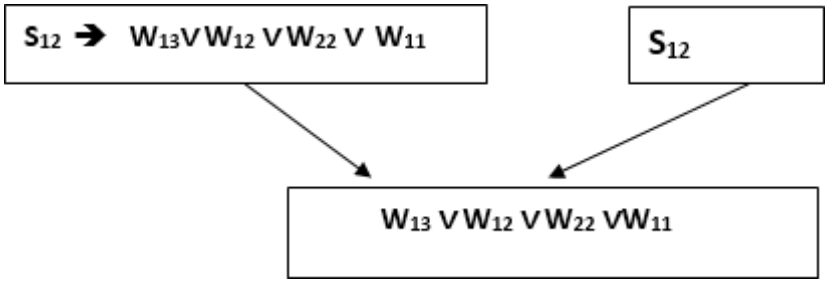
Apply And -Elimination rule:

Now again apply And-elimination rule to ¬ W21 ∧ ¬ W22 ∧¬ W31, We will get three statements:

¬ W21, ¬ W22, and ¬ W31.

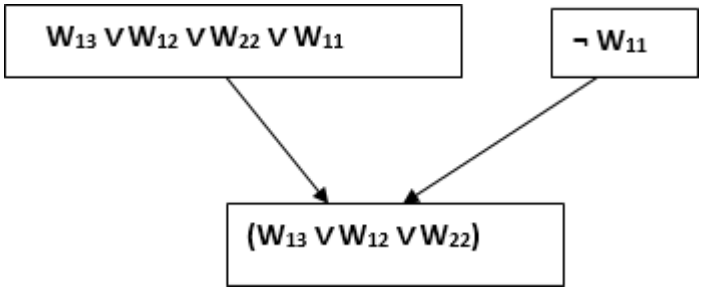
Apply MP to S12 and R4:

Apply Modus Ponens to S12 and R4 which is S12 → W13 ∨. W12 ∨. W22 ∨.W11, we will get the output as W13∨ W12 ∨ W22 ∨.W11.



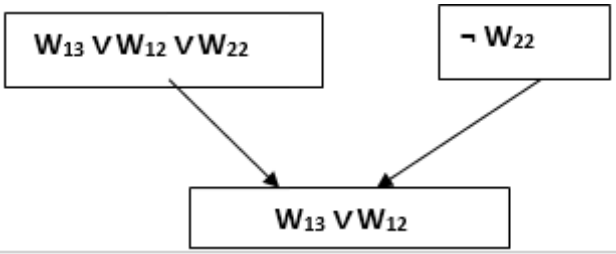
Apply Unit resolution on W13 ∨ W12 ∨ W22 ∨W11 and ¬ W11 :

After applying Unit resolution formula on W13 ∨ W12 ∨ W22 ∨W11 and ¬ W11 we will get W13 ∨ W12 ∨ W22.



Apply Unit resolution on W13 ∨ W12 ∨ W22 and ¬ W22 :

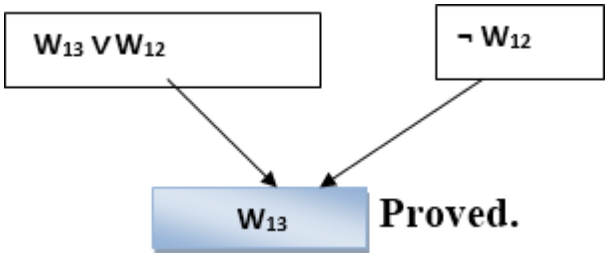
After applying Unit resolution on W13 ∨ W12 ∨ W22, and ¬W22, we will get W13 ∨ W12 as output.



Apply Unit Resolution on W13 ∨ W12 and ¬ W12 :

After Applying Unit resolution on W13 ∨ W12 and ¬ W12, we will get W13 as an output,

hence it is proved that the Wumpus is in the room [1, 3].



5.	**Interpret in detail about FOL inference rules for quantifier with an example.**

As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

Universal Generalization
Universal Instantiation
Existential Instantiation
Existential introduction

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: $\frac{P(c)}{\forall x\, P(x)}$

.
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.

Universal Instantiation:

Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.
The new KB is logically equivalent to the previous KB.
As per UI, we can infer any sentence obtained by substituting a ground term for the variable.
The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ x P(x) for any object in the universe of discourse.

It can be represented as: $\frac{\forall x\, P(x)}{P(c)}$.
Example:1.
IF "Every person like ice-cream"=> ∀x P(x) so we can infer that "John likes ice-cream" => P(c)
Example: 2.
Let's take a famous example,
"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:
∀x king(x) ∧ greedy (x) → Evil (x),

	So from this information, we can infer any of the following statements using Universal **Instantiation:** King(John) ∧ Greedy (John) → Evil (John), King(Richard) ∧ Greedy (Richard) → Evil (Richard), King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)), **Existential Instantiation:** Existential instantiation 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: $\frac{\exists x\, P(x)}{P(c)}$ 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. **Existential introduction** An existential introduction 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: $\frac{P(c)}{\exists x P(x)}$ Example: Let's say that, "Priyanka got good marks in English." "Therefore, someone got good marks in English."
6.	**Explain in detail about forward chaining along with its proof?** 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.

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

Properties of Forward-Chaining:

- 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.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Consider the following famous example which we will use in both approaches:

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."

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts Conversion into FOL:

It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

Country A has some missiles. ?p Owns(A, p) ∧ Missile(p). It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

Owns(A, T1) ... (2)

Missile(T1) ... (3)

All of the missiles were sold to country A by Robert.

?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) (4)

Missiles are weapons.

Missile(p) → Weapons (p) (5)

Enemy of America is known as hostile.

Enemy(p, America) →Hostile(p) (6)

Country A is an enemy of America.

Enemy (A, America) (7)

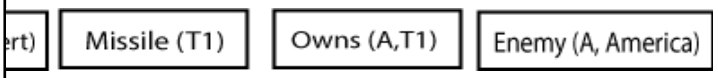
Robert is American

American(Robert) (8)

Forward chaining proof:

Step-1:

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1). All these facts will be represented as below.



Step-2:

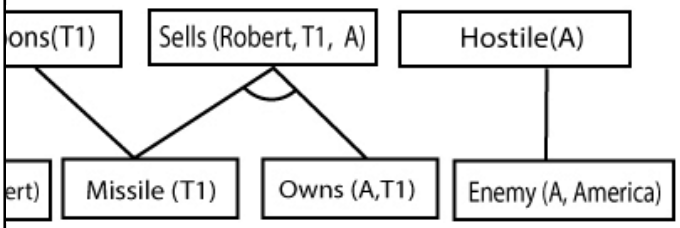
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

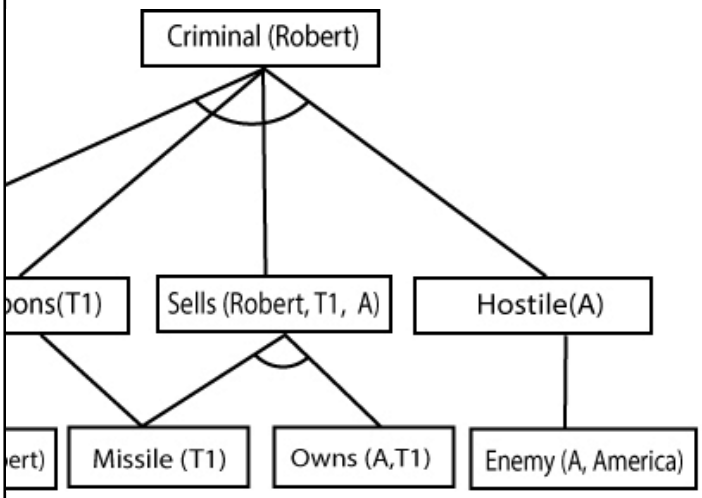
Rule-(4) satisfy with the substitution {p/T1}, so Sells (Robert, T1, A) is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



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. And hence we reached our goal statement.

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

7

Explain in detail about backward chaining along with its proof?

Backward-chaining is also known as a backward deduction or backward reasoning 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.

Properties of backward chaining:

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-goals 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.

The backward-chaining method mostly used a depth-first search strategy for proof.

Example:

In backward-chaining, we will use the same above example, and will rewrite all the rules.

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

Owns(A, T1) (2)

Missile(T1)

?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) (4)

Missile(p) → Weapons (p) (5)

Enemy(p, America) →Hostile(p) (6)

Enemy (A, America) (7)

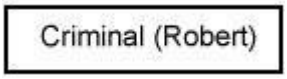
American(Robert) (8)

Backward-Chaining proof:

In Backward 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 of it.

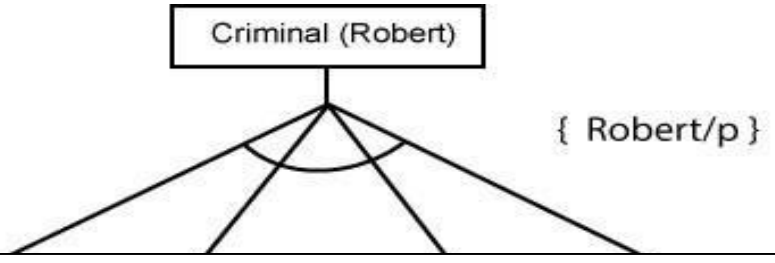


Step-2:

At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution

{Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.

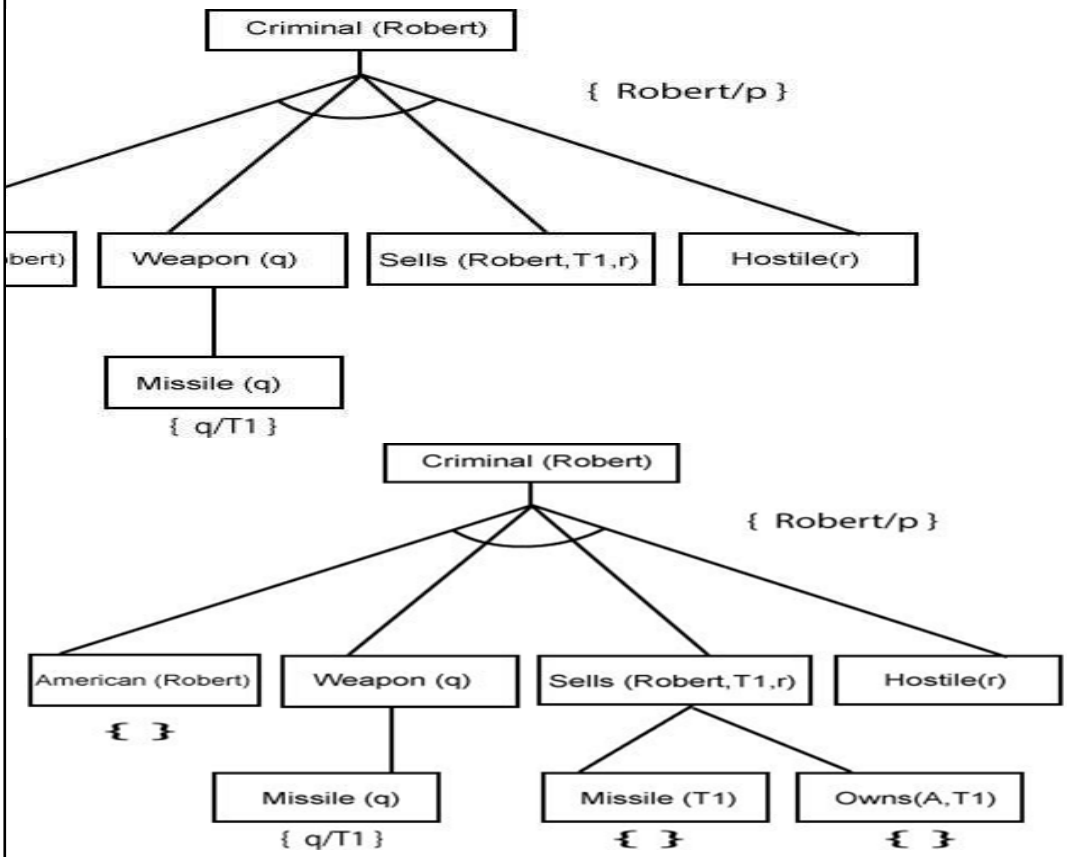
Here we can see American (Robert) is a fact, so it is proved here.



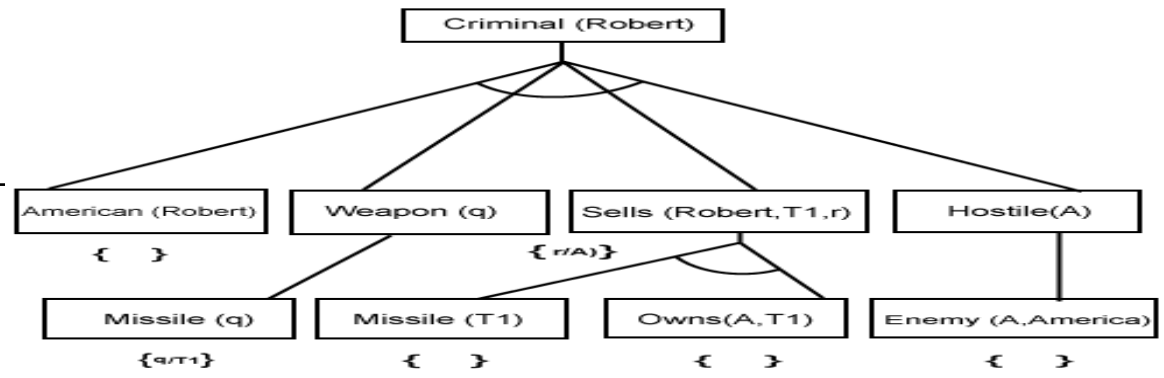
Step-3:t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.

Step-4:

At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the Rule- 4, with the substitution of A in place of r. So these two statements are proved here.



Step-5: In step 5, we can infer the fact Enemy(A, America) from Hostile(A), which satisfies Rule 6. Hence, all the statements are proven true using backward chaining.



8	**Solve the below given problem using resolution FOL** (x) → likes(John, x) ple) ∧ food(vegetables) ats(x, y) ∧ ¬ killed(x) → food(y) l, Peanuts) ∧ alive(Anil). (Anil, x) → eats(Harry, x) ed(x) → alive(x) } **added predicates.** e(x) →¬ killed(x) n, Peanuts) **<u>Answers</u>** **Step1 is already mentioned above the facts are converted into FOL** **Step-1: Conversion of Facts into FOL** a. ∀x : food(x) → likes(John, x) b. food(Apple) ∧ food(vegetables) c. ∀x ∀y : eats(x, y) ∧ ¬ killed(x) → food(y) d. eats (Anil, Peanuts) ∧ alive(Anil) e. ∀x : eats(Anil, x) → eats(Harry, x) f. ∀x: ¬ killed(x) → alive(x) } **added predicates** g. ∀x : alive(x) → ¬ killed(x) h. likes(John, Peanuts). Step-2: Conversion of FOL into CNF In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

Step-2: Conversion of FOL into CNF

2.1 Eliminate all implication (→) and rewrite

a. ∀x ¬ food(x) V likes(John, x) **P—>Q= ¬P V Q**

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x¬ [¬ killed(x)] V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).

Step-2: Conversion of FOL into CNF

2.2 Move negation (¬)inwards and rewrite

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x killed(x) V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).

Step-2: Conversion of FOL into CNF

2.3 Rename variables or standardize variables

a. ∀x ¬ food(x) V likes(John, x)
b. food(Apple) Λ food(vegetables)
c. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
d. eats (Anil, Peanuts) Λ alive(Anil)
e. ∀w¬ eats(Anil, w) V eats(Harry, w)
f. ∀g killed(g) V alive(g)
g. ∀k ¬ alive(k) V ¬ killed(k)
h. likes(John, Peanuts).

Eliminate existential instantiation quantifier by elimination.
In this step, we will eliminate existential quantifier ∃, and this process is known as Skolemization. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

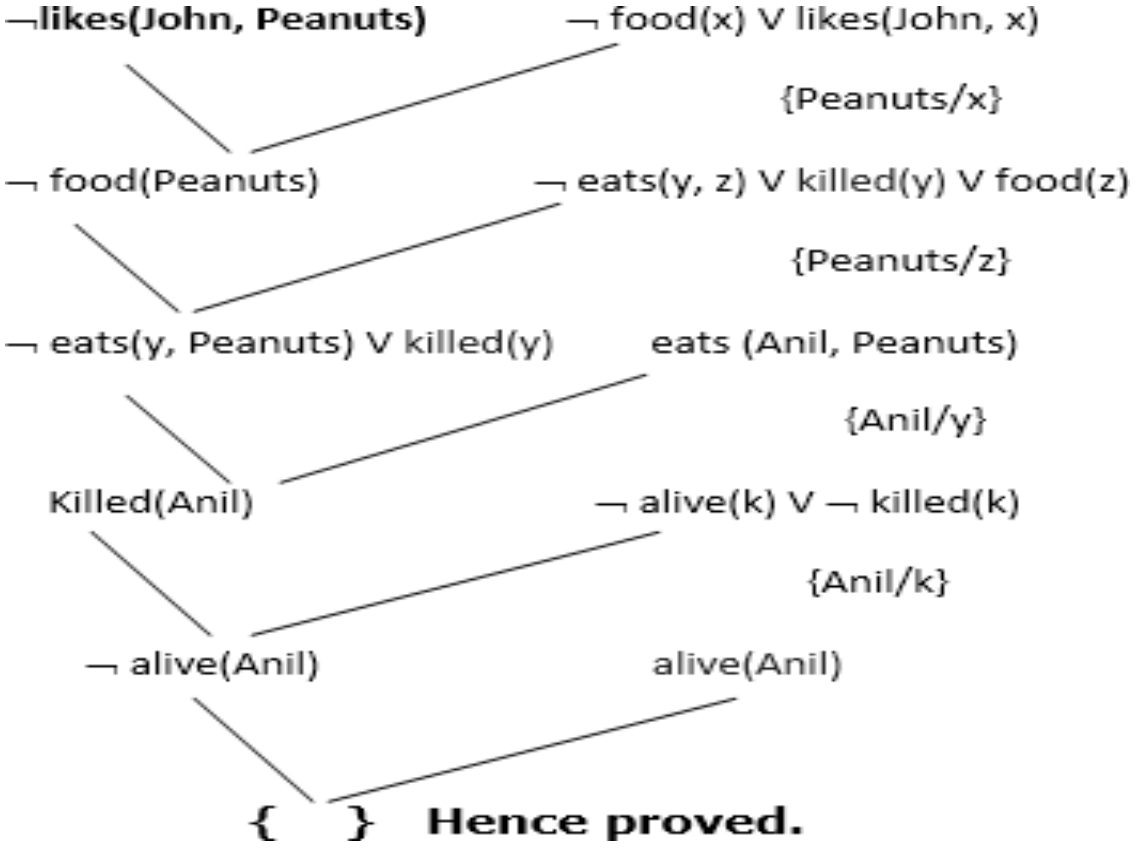
Step-2: Conversion of FOL into CNF

2.4 Eliminate existential instantiation quantifier by elimination.

2.5 Drop Universal quantifiers.

a. ¬ food(x) V likes(John, x)
b. food(Apple)
c. food(vegetables)
d. ¬ eats(y, z) V killed(y) V food(z)
e. eats (Anil, Peanuts)
f. alive(Anil)
g. ¬ eats(Anil, w) V eats(Harry, w)
h. killed(g) V alive(g)
i. ¬ alive(k) V ¬ killed(k)
j. likes(John, Peanuts).

2.6 Distribute conjunction Λ over disjunction ¬.

	ments "food(Apple) Λ food(vegetables)" and "eats (Anil, alive(Anil)" can be written in two separate statements. Distribute conjunction Λ over disjunction ¬. This step will not make any change in this problem. **Step-3: Negate the statement to be proved** In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts) **Step-4: Draw Resolution graph:** Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows: 

Q.No	Part - C
1.	**Interpret in detail about Applications of Bayesian Network in AI.** **Answers:** Bayes' Rule is used in various AI applications, particularly in situations where we need to reason about uncertainty and make decisions under uncertainty. Below are some of the key applications in AI: **1. Bayesian Inference in Machine Learning** Bayesian Inference refers to the process of using Bayes' Rule to update the probability of a model or hypothesis as more data becomes available. It allows for the development of

probabilistic models where uncertainty is explicitly modeled and updated over time.

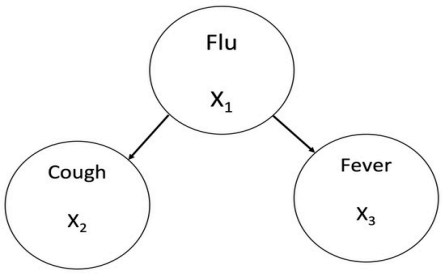
Example: In a classification task (e.g., spam email detection), we can use Bayes' Rule to compute the probability that an email is spam based on observed features (e.g., presence of certain words, frequency of certain terms). As new evidence (features of a new email) comes in, we update the probability that the email belongs to the spam class using the prior (based on training data) and the likelihood (probability of seeing those words given the class).

Naive Bayes Classifier is one of the most popular machine learning algorithms based on Bayes' Theorem. It assumes that the features are conditionally independent given the class label, making it computationally efficient.

2. Bayesian Networks

- A Bayesian Network is a directed acyclic graph (DAG)
- Nodes represent random variables (e.g., features, attributes, or states in a system).
- Edges (arcs) represent probabilistic dependencies or conditional dependencies between these variables.
- The main idea behind a Bayesian Network is that each node in the graph is conditionally independent of its non-descendants, given its parents in the network.
- This allows the model to represent complex joint distributions using a much smaller number of parameters than would be needed to represent the full distribution explicitly.

A simple Bayesian Network for diagnosing a disease



Flu is expected to causally influence the presence of a cough or fever.

if a person has the flu, it is reasonable to conclude that the presence of a fever does not depend on the presence of a cough. Therefore, we assume that the variables cough ($X2$) and fever ($X3$) are conditionally independent given the variable flu ($X1$)

Formally, this can be expressed as follows:

$P(X2|X1,X3)=P(X2|X1)$

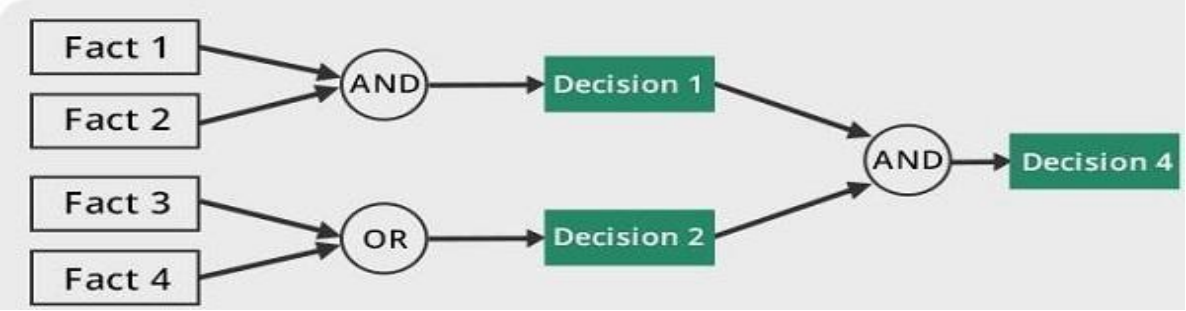
$P(X3|X1,X2)=P(X3|X1)$.

Therefore,

$P(X1,X2,X3)=P(X2|X1)P(X3|X1)$.

Applications of Bayesian Networks:

	Medical Diagnosis: Bayesian networks can model the relationships between symptoms, diseases, and treatments. They can be used to diagnose diseases based on observed symptoms and provide probabilities for the presence of specific diseases. **Risk Analysis:** In fields like finance and engineering, Bayesian networks can model the dependencies between various risk factors (e.g., market conditions, company performance) to assess the likelihood of different outcomes. **Decision Making Under Uncertainty** Bayes' Rule is crucial for decision-making in uncertain environments, where we may have incomplete or noisy information. By updating probabilities as new evidence becomes available, Bayes' Rule helps make optimal decisions. Example: In autonomous driving, a vehicle might use sensor data (e.g., radar, camera) to update its belief about its surroundings (e.g., the position of pedestrians, other vehicles). Bayes' Rule helps the vehicle compute the probability distribution over possible scenarios and select the best action (e.g., stop, accelerate, or steer). **Spam Filtering** Spam filtering is a classic example of applying Bayes' Rule for classification tasks, especially in email systems. Naive Bayes, a simple probabilistic classifier based on Bayes' Rule, is commonly used for spam detection. Process: Given a set of training emails labeled as spam or not spam, the Naive Bayes classifier computes the probability of an email being spam based on the frequency of certain words in the email (e.g., "win", "free", "discount"). Bayes' Rule is used to compute the posterior probability of the email being spam, given the observed words. The email is classified as spam if the posterior probability exceeds a certain threshold. **Bayesian Optimization** Bayesian optimization is a method used to optimize expensive or noisy functions, typically in hyperparameter tuning for machine learning models or optimization of complex systems. It builds a probabilistic model of the objective function and uses Bayes' Rule to guide the search for the optimal solution. **Application:** In hyperparameter tuning for machine learning models, Bayesian optimization can be used to efficiently search the hyperparameter space by modeling the performance of different configurations and choosing the next hyperparameters to test based on the current posterior belief. **Example: Bayesian Spam Classifier (Naive Bayes)** To understand how Bayes' Rule works in a real-world application, consider a Naive Bayes classifier for spam detection. The goal is to classify an email as spam or not spam based on the presence of certain words.
2.	**Explain in detail about forward and backward chaining with appropriate example.** **Answer:** 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.



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

Properties of Forward-Chaining:

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.

Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

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."

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts Conversion into FOL:

It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

Country A has some missiles. ?p Owns(A, p) ∧ Missile(p). It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

Owns(A, T1) ... (2)

Missile(T1) ... (3)

All of the missiles were sold to country A by Robert.

?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) (4)

Missiles are weapons.

Missile(p) → Weapons (p) (5)

Enemy of America is known as hostile.

Enemy(p, America) →Hostile(p) (6)

Country A is an enemy of America.

Enemy (A, America) (7)

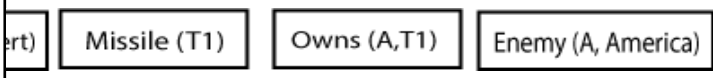
Robert is American

American(Robert) (8)

Forward chaining proof:

Step-1:

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1). All these facts will be represented as below.



Step-2:

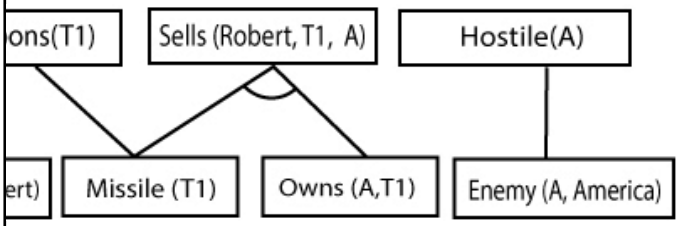
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

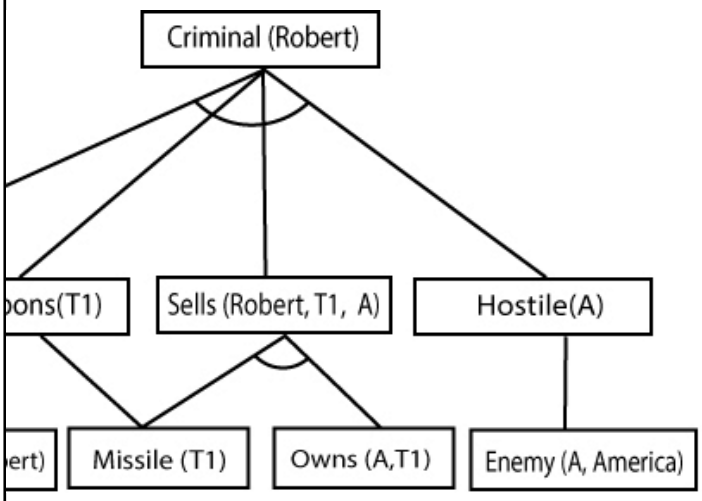
Rule-(4) satisfy with the substitution {p/T1}, so Sells (Robert, T1, A) is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



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

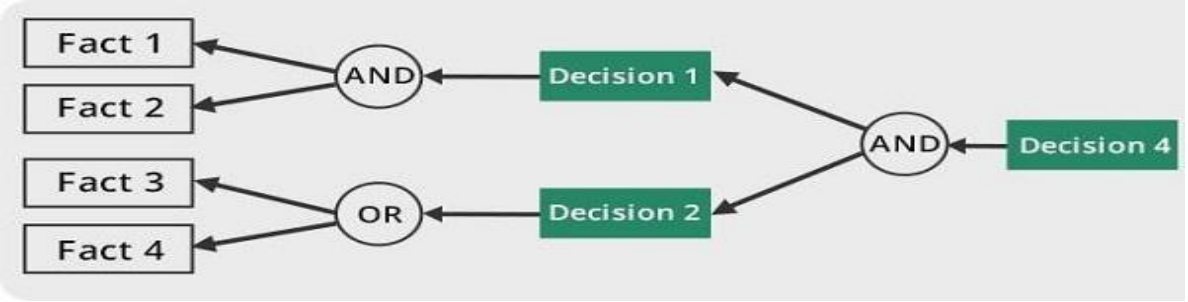


Criminal(Robert) which infers all the available facts. And hence we reached our goal statement.

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

Backward-chaining is also known as a backward deduction or backward reasoning 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.



Properties of backward chaining:

- 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-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

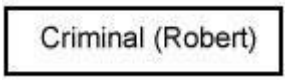
Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

The backward-chaining method mostly used a depth-first search strategy for proof.

In Backward 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 of it.

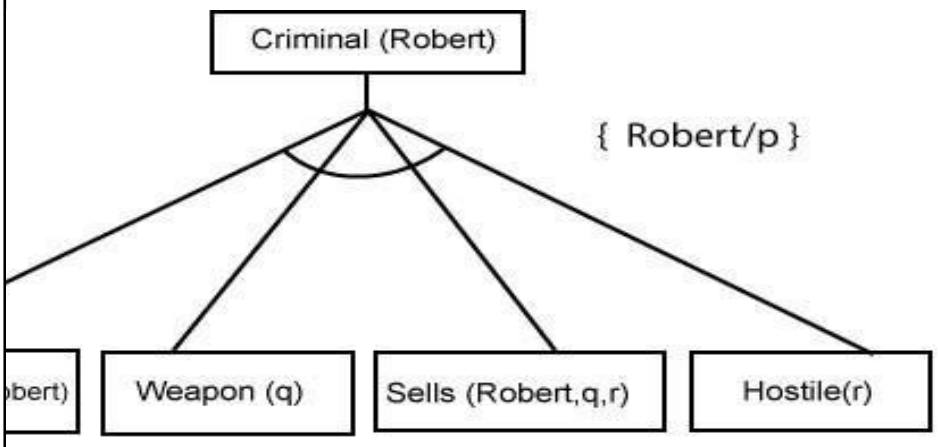


Step-2:

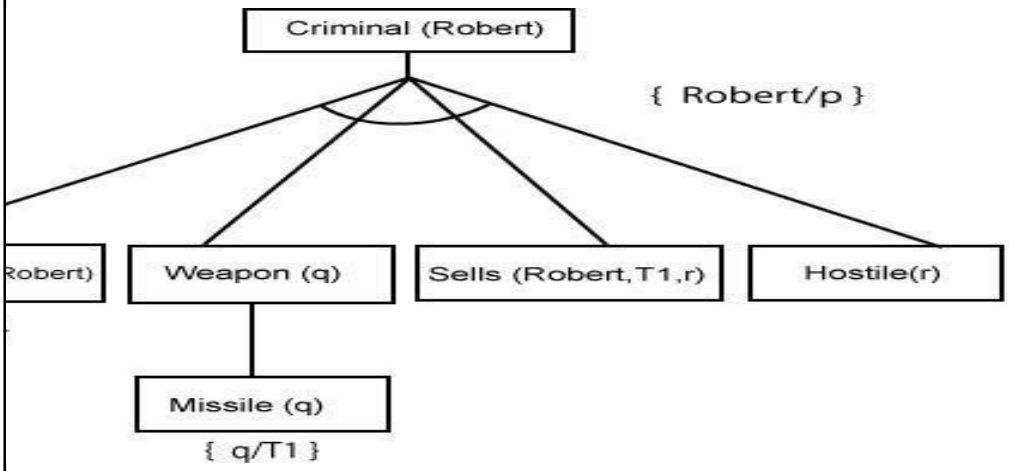
At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution

{Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.

Here we can see American (Robert) is a fact, so it is proved here.

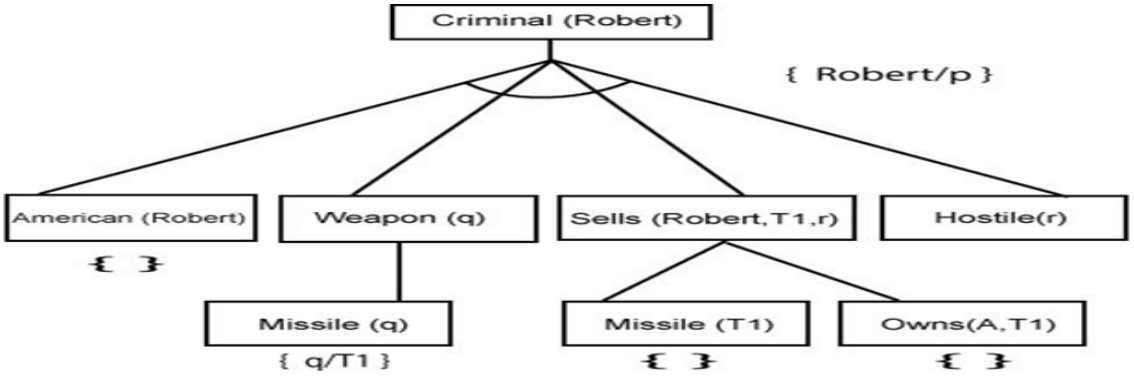


Step-3:t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



Step-4:

At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the Rule- 4, with the substitution of A in place of r. So these two statements are proved here.



Step-5: In step 5, we can infer the fact Enemy(A, America) from Hostile(A), which satisfies Rule 6. Hence, all the statements are proven true using backward chaining.