

Q1. dumpster shafer theory

The **Dumpster-Shafer Theory**, more commonly referred to as **Dempster-Shafer Theory (DST)**, is a mathematical framework for reasoning with uncertainty, which is widely used in fields like artificial intelligence (AI) and decision-making systems. Here's a summary:

Key Concepts of Dempster-Shafer Theory:

1. Belief Functions:

- Unlike probability theory, DST works with degrees of belief rather than fixed probabilities.
- It allows representation of uncertainty and ignorance.

2. Frame of Discernment (Θ):

- The set of all possible outcomes or hypotheses in a given situation.
- For example, in a medical diagnosis scenario, Θ could be { Disease A, Disease B, Disease C }.

3. Basic Probability Assignment (BPA):

- Assigns a "belief mass" to subsets of Θ , including the empty set.
- The BPA, denoted as $m(A)$, must satisfy:
 - $0 \leq m(A) \leq 1$ for all subsets AA of Θ .
 - $\sum_{A \subseteq \Theta} m(A) = 1$.

4. Belief (Bel) and Plausibility (Pl):

- **Belief (Bel):** The total belief committed to a set AA, considering all evidence that supports AA.
- **Plausibility (Pl):** Represents the extent to which evidence does not refute AA.
- $Bel(A) \leq Pl(A)$, where the gap between them reflects uncertainty.

5. Dempster's Rule of Combination:

- A mechanism for combining evidence from different sources.
- It combines two BPAs m_1 and m_2 to produce a new BPA m :
$$m(C) = \sum_{A \cap B = C} m_1(A) \cdot m_2(B) / \sum_{A \cap B \neq \emptyset} m_1(A) \cdot m_2(B)$$
- This rule ignores conflicting evidence by normalizing the denominator.

Applications in AI:

- **Sensor Fusion:** Combining data from multiple sensors to make a decision (e.g., autonomous vehicles).
- **Expert Systems:** Handling uncertain or incomplete knowledge in decision-making.
- **Medical Diagnosis:** Integrating uncertain evidence from tests and symptoms.
- **Risk Assessment:** Managing uncertainty in scenarios like financial markets or engineering systems.

The Dempster-Shafer Theory is particularly useful where probabilistic data is unavailable or incomplete, allowing for reasoning about uncertainty without requiring precise probabilities.

Limitations of DST

1. Computational Complexity:

- Combining evidence is computationally expensive, especially for large frames of discernment.

2. Conflict Resolution:

- High conflict between evidence sources can lead to counterintuitive results.

3. Interpretation Challenges:

- Belief and plausibility functions may be harder to interpret than probabilities.

Q2. convert statement into logic

Examples:

Example 1: Propositional Logic

- **Statement:** "If it rains, then the ground will be wet."
- **Logic:** $R \rightarrow W$
 - Where R = "It rains" and W = "The ground is wet."

Example 2: Predicate Logic

- **Statement:** "All humans are mortal."
- **Logic:** $\forall x(\text{Human}(x) \rightarrow \text{Mortal}(x))$
 - Where $\text{Human}(x)$ = "x is a human" and $\text{Mortal}(x)$ = "x is mortal."

Example 3: Fuzzy Logic

- **Statement:** "The weather is warm."
- **Logic:** $\text{Warm}(\text{Weather})=0.7$
 - Assigning a degree of truth (e.g., 70%) to the predicate $\text{Warm}(\text{Weather})$.

Example 4: Combining Logic Types

- **Statement:** "If a person studies hard and is intelligent, then they will succeed."
- **Logic:** $\forall x(\text{StudiesHard}(x) \wedge \text{Intelligent}(x) \rightarrow \text{Succeeds}(x))$
 - Combining multiple conditions with \wedge (AND) and implication \rightarrow .

Converting Statements into Logic

When converting natural language statements into logic, you can choose a framework based on the level of detail and expressiveness required:

1. Propositional Logic:

- Suitable for **simple statements** without involving specific objects or relationships.
- Example: "It is raining" $\rightarrow R$.
- **No variables** or relationships are used.

2. First-Order Logic (FOL):

- Suitable for **complex statements** involving objects, relationships, and quantifiers.

- Example: "All humans are mortal" $\rightarrow \forall x(\text{Human}(x) \rightarrow \text{Mortal}(x))$.
- **FOL is particularly useful for converting statements with generalizations or relationships** (e.g., "A loves B").

3. Other Logical Systems:

- **Fuzzy Logic:** Used for vague or uncertain statements (e.g., "It is somewhat warm").
 - **Modal Logic:** Captures necessity and possibility (e.g., "It is necessary to breathe").
 - **Temporal Logic:** Represents time-dependent statements (e.g., "Eventually it will rain").
-

Is First-Order Logic Always Used?

Not necessarily. FOL is a **specific tool** in the broader process of converting statements into logic. Whether FOL is used depends on the statement's complexity:

- **Simple Statements:** Propositional logic may suffice.
 - "If the light is on, then the room is bright."
 - Logic: L \rightarrow B.
 - **Relational or Quantified Statements:** First-Order Logic is required.
 - "Everyone who studies hard will succeed."
 - Logic: $\forall x(\text{StudiesHard}(x) \rightarrow \text{Succeeds}(x))$.
-

When to Use First-Order Logic

1. **Relationships:** When the statement involves a relationship between objects (e.g., "John loves Mary").
 2. **Quantifiers:** When general rules or specific existence statements are involved (e.g., "All humans are mortal").
 3. **Inference:** When reasoning about a domain where new facts need to be inferred (e.g., "If John is mortal and John is a human, then all humans are mortal").
-

Example: Converting a Statement into FOL

- **Statement:** "Some students in the class are excellent."
- **FOL Representation:**
 - $\exists x(\text{Student}(x) \wedge \text{Excellent}(x))$.
 - Explanation:
 - $\exists x$: There exists at least one x.
 - Student(x): x is a student.
 - Excellent(x): x is excellent.

Q3.What is forward chaining and backward chainng

Forward Chaining and backward chaining in AI

In artificial intelligence, forward and backward chaining is one of the important topics, but before understanding forward and backward chaining lets first understand that from where these two terms came.

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

A. 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 starts 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 adds 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) \wedge weapon(q) \wedge sells(p, q, r) \wedge hostile(r) \rightarrow Criminal(p)(1)
- Country A has some missiles. ?p Owns(A, p) \wedge 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) \wedge Owns(A, p) \rightarrow Sells(Robert, p, A)(4)
- Missiles are weapons.
Missile(p) \rightarrow Weapons(p)(5)
- Enemy of America is known as hostile.
Enemy(p, America) \rightarrow 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:

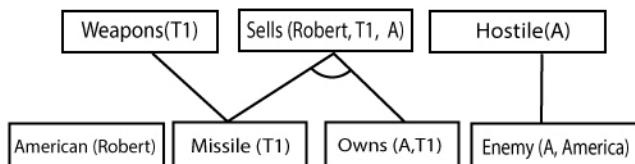
At the second step, we will see those facts which infer from available facts and with satisfied premises.

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

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

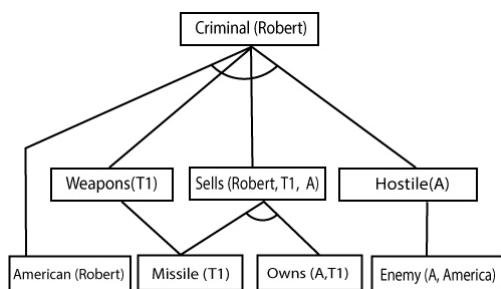
Rule-(4) satisfy with the substitution { p/T1 }, so **Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

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.

B. Backward Chaining:

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.

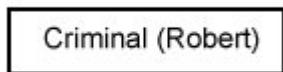
- American (p) \wedge weapon(q) \wedge sells (p, q, r) \wedge hostile(r) \rightarrow Criminal(p) ... (1)
 - Owns(A, T1)(2)
 - Missile(T1)
 - ?p Missiles(p) \wedge Owns (A, p) \rightarrow Sells (Robert, p, A)(4)
 - Missile(p) \rightarrow Weapons (p)(5)
 - Enemy(p, America) \rightarrow Hostile(p)(6)
 - 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:

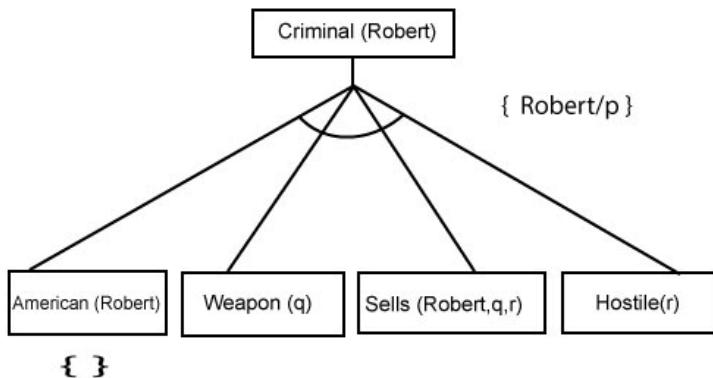
At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.



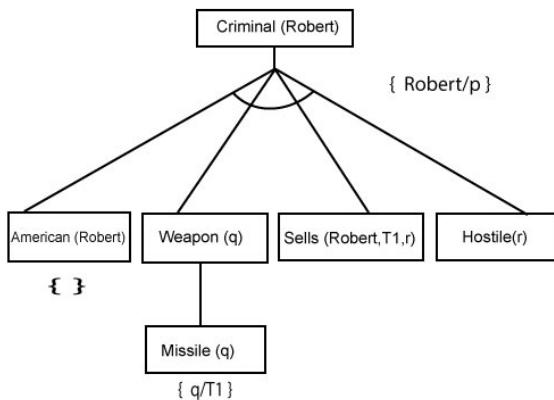
Step-2:

At the second step, we will infer other facts from 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.

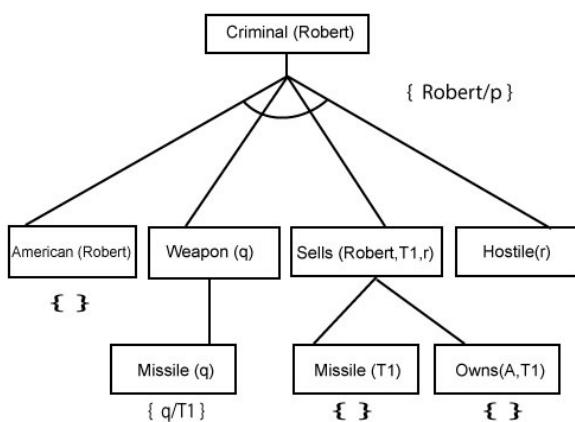


Step-3:t At step-3, we will extract further fact $\text{Missile}(q)$ which infer from $\text{Weapon}(q)$, as it satisfies Rule-(5). $\text{Weapon}(q)$ is also true with the substitution of a constant $T1$ at q .



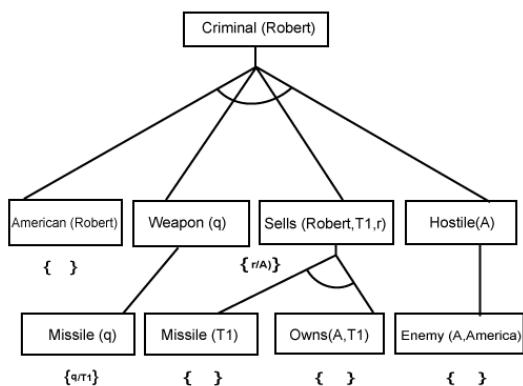
Step-4:

At step-4, we can infer facts **Missile(T1)** and **Owns(A, T1)** from **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:

At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.



Q4. What is an Expert System?

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

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using **both facts and**

heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science, etc.**

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

Below are some popular examples of the Expert System:

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

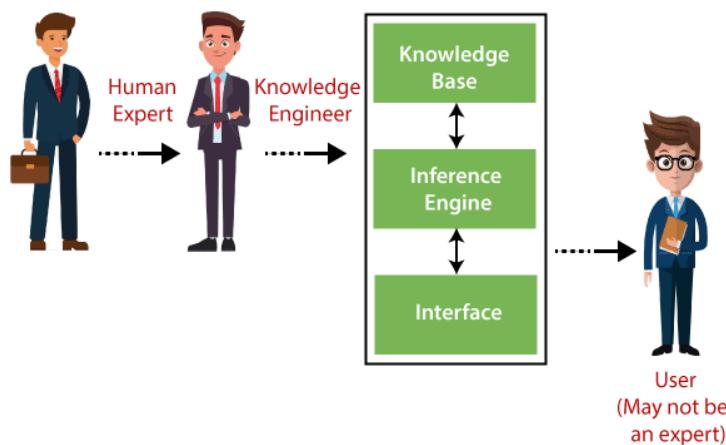
Characteristics of Expert System

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

Components of Expert System

An expert system mainly consists of three components:

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



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the

output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

2. Inference Engine(Rules of Engine)

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- There are two types of inference engine:
- Deterministic Inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts and rules**.
- Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- Forward Chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- Backward Chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. Knowledge Base

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

Factual Knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

- Heuristic Knowledge:** This knowledge is based on practice, the ability to guess, evaluation, and experiences.

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

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

Advantages of Expert System

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

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.
- Its maintenance and development costs are very high.

- Knowledge acquisition for designing is much difficult.
- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

Applications of Expert System

• In designing and manufacturing domain

It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.

• In the knowledge domain

These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.

• In the finance domain

In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.

• In the diagnosis and troubleshooting of devices

In medical diagnosis, the ES system is used, and it was the first area where these systems were used.

• Planning and Scheduling

The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

Q5. Decision tree example

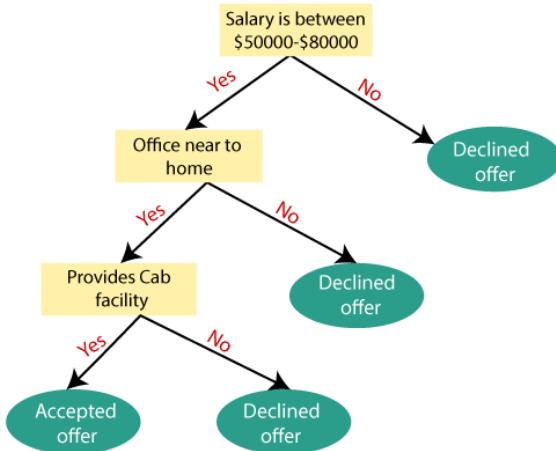
Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome**.

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node**. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm**, which stands for **Classification and Regression Tree algorithm**.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.

Decision Tree Terminologies

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

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



Q6.Resolution techniques

Resolution Techniques in AI are methods used to perform automated reasoning in logical systems, particularly in **Propositional Logic** and **First-Order Logic (FOL)**. Resolution is a powerful inference method widely used in theorem proving, problem-solving, and logic-based AI systems. Here's an overview of **resolution techniques** and their application in AI:

What is Resolution?

- **Resolution** is a rule of inference used to derive new facts from a set of given facts (clauses) in logic.
- It works by combining two clauses with complementary literals to produce a new clause (called the **resolvent**).
- Resolution is based on the **Principle of Refutation**, meaning it tries to derive a contradiction to prove the unsatisfiability of a set of clauses.

Steps in Resolution Technique

1. Convert to Clausal Form:

- Any logical formula must first be converted into **Conjunctive Normal Form (CNF)**, which is a conjunction of disjunctions of literals.
- Example:
 - $(A \wedge B) \rightarrow C$ becomes $(\neg A \vee \neg B \vee C)$.

2. Negate the Goal:

- To prove G, assume $\neg G$ and try to derive a contradiction.

3. Apply the Resolution Rule:

- Identify pairs of clauses that contain complementary literals (e.g., P and $\neg P$).
- Combine these clauses into a new resolvent by removing the complementary literals.

4. Iterate Until a Contradiction is Found:

- If a contradiction (empty clause) is derived, the original set of clauses is unsatisfiable, proving the goal.

5. Conclude:

- If a contradiction is found, the original statement is true.
- If no contradiction is found, the statement cannot be proven.

Resolution Rule

The resolution rule states:

- From two clauses ($A \vee P$) and ($\neg P \vee B$), derive ($A \vee B$).
 - Here, P and $\neg P$ are complementary literals.
-

Types of Resolution Techniques

1. Propositional Resolution:

- Applies to propositional logic.
- Example:
 - Clauses: $(P \vee Q)$, $(\neg P \vee R)$.
 - Resolvent: $(Q \vee R)$.

2. First-Order Resolution:

- Extends resolution to First-Order Logic (FOL) by incorporating **unification**.
- **Unification:**
 - A process of making two logical expressions identical by substituting variables.
- Example:
 - Clauses: $\text{Loves}(\text{John}, x) \vee \text{Hates}(\text{John}, x)$, $\neg \text{Loves}(\text{John}, \text{Mary})$.
 - Resolvent: $\text{Hates}(\text{John}, \text{Mary})$.

3. Unit Resolution:

- Simplifies resolution by focusing on a single literal clause.
- Example:
 - (P) , $(\neg P \vee Q)$.
 - Resolvent: Q .

4. Input Resolution:

- Always resolves one clause from the original set of clauses with others.
- Reduces the search space in resolution.

5. Linear Resolution:

- Resolves a chain of clauses derived from a single goal, ensuring a linear proof sequence.

6. Set of Support Resolution:

- Only resolves clauses with a specific "set of support," typically the goal and its derived clauses.
 - Ensures focus on relevant parts of the knowledge base.
-

Applications in AI

1. Automated Theorem Proving:

- Prove mathematical theorems or logical statements using resolution.
- Example: Proving $\forall x(P(x) \rightarrow Q(x))$.

2. Logic Programming:

- Used in systems like **Prolog** to infer logical conclusions.

3. Knowledge Representation and Reasoning:

- Derive new knowledge from a set of rules and facts.

4. Natural Language Processing (NLP):

- Logical resolution can help in semantic understanding and reasoning tasks.

5. Expert Systems:

- Derive conclusions in rule-based systems through logical inference.
-

Example: Proof by Resolution

Problem:

- Prove $(A \vee B)$ is true from the following:

1. $\neg A \vee B$
2. $\neg B \vee A$

Steps:

1. Negate the goal: Assume $\neg(A \vee B) = \neg A \wedge \neg B$.
 - This gives two new clauses: $\neg A$ and $\neg B$.
2. Combine clauses:
 - $(\neg A)$ and $(\neg A \vee B)$ resolve to B .
 - $(\neg B)$ and $(\neg B \vee A)$ resolve to A .
3. Contradiction:
 - B contradicts $\neg B$, and A contradicts $\neg A$.
4. Conclusion:
 - The original statement $(A \vee B)$ is true.

Advantages and Limitations

Advantages:

- Simple and systematic.
- Applicable to propositional and first-order logic.
- Widely used in AI reasoning systems.

Limitations:

- Can be computationally expensive (exponential complexity).
- Requires conversion to CNF, which can lead to an explosion of clauses.
- May not always terminate in FOL due to undecidability.