Code No: 155DY **R18**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year I Semester Examinations, January/February - 2023
ARTIFICIAL INTELLIGENCE
(Common to CESE, CSE(CS), CSE(DS))

Time: 3 Hours **Max. Marks: 75**

Note: i) Question paper consists of Part A and Part B.
ii) Part A is compulsory, which carries 25 marks. In Part A, Answer all questions.
iii) In Part B, Answer any one question from each unit. Each question carries 10 marks and may have a, and b as sub-questions.

PART – A

(25 Marks)

1. a) What is intelligent agent? [2]
b) Explain about uniform cost search. [3]
c) What is constraint propagation? [2]
d) Explain about alpha-beta pruning. [3]
e) What is backward chaining? [2]
f) Differentiate between unification and lifting. [3]
g) Give the definition of classical planning. [2]
h) Explain about multi-agent planning. [3]
i) What is uncertain knowledge? [2]
j) What are the different forms of learning? [3]

PART – B

(50 Marks)

2.a) Explain about greedy best-first search technique.
b) What simulated annealing search? Explain. [5+5]

OR

3.a) Discuss about A* search and Bidirectional search techniques.
b) Explain in detail about online search agents. [5+5]

4.a) What are knowledge-based agents? How can a knowledge-based agent be described at three levels?
b) Give ontological and epistemological commitments of a propositional logic. [5+5]

OR

5.a) Define the constraint satisfaction problem and explain the backtracking search for CSPs. [5+5]
b) What is Adversarial Search? Explain adversarial search techniques.

6.a) Write about resolution in first-order logic.
b) What is Ontological engineering? Explain. [5+5]

OR

7.a) Explain the semantics of first-order logic in knowledge representation.
b) Explain about backward chaining in first-order logic. [5+5]

8. List and explain different classical planning approaches. [10]

OR

9.a) Explain forward state space search with an example.
b) Discuss about hierarchical planning with an example. [5+5]

10.a) Discuss in brief about Dempster-Shafer theory.
b) How to represent knowledge in an uncertain domain? Explain. [5+5]

OR

11.a) Explain Bayes' rule and its uses.
b) Discuss in brief about explanation-based learning. [5+5]

---ooOoo---

ANSWER KEY

PART – A

1.a) What is an intelligent agent?

An intelligent agent is a software or hardware entity designed to observe its environment, learn from experience, and autonomously take actions to maximize the likelihood of achieving predefined goals. These agents can operate in a variety of contexts, from simple programs to complex robotic systems.

b) Explain uniform cost search.

Uniform Cost Search is a graph search algorithm used to find the least-cost path from a start node to a goal node in a weighted graph. It maintains a priority queue to expand nodes with the lowest cumulative cost first, ensuring an optimal solution when all edge costs are non-negative.

c) What is constraint propagation?

Constraint propagation is a technique in artificial intelligence where the possible values of variables are iteratively narrowed down based on the constraints imposed on them. This process helps eliminate inconsistent or infeasible values, aiding in problem-solving scenarios with interdependent variables.

d) Explain alpha-beta pruning.

Alpha-beta pruning is an optimization technique applied in the minimax algorithm for game tree search. By intelligently discarding branches that cannot affect the final decision, it significantly reduces the number of nodes evaluated, improving the efficiency of the search process in games like chess.

e) What is backward chaining

Backward chaining is an inference strategy where the system starts with a goal and works backward to determine the sequence of actions or events needed to achieve that goal. This approach is commonly used in rule-based systems and expert systems for decision-making and problem-solving.

f) Differentiate between unification and lifting.

Unification is the process of finding a common substitution for two expressions, making them identical. On the other hand, lifting involves applying operations or functions to entire sets or sequences of values, introducing a level of abstraction beyond individual elements.

g) Give the definition of classical planning.

Classical planning involves determining a sequence of actions to achieve a goal in a deterministic, fully observable environment with a known initial state. This

approach is often applied in scenarios where the outcomes of actions are certain and predictable.

h) Explain multi-agent planning.
Multi-agent planning addresses the coordination of actions among multiple intelligent agents to achieve shared goals. This field considers the complexities arising from interactions, dependencies, and conflicting interests between agents, requiring sophisticated planning strategies.

i) What is uncertain knowledge?
Uncertain knowledge refers to information characterized by inherent ambiguity or imprecision, often involving probabilities or degrees of belief. In the context of artificial intelligence, dealing with uncertain knowledge is crucial for decision-making in environments where outcomes are not entirely predictable.

j) What are the different forms of learning?
The different forms of learning include supervised learning, where models are trained on labeled data; unsupervised learning, which involves discovering patterns in unlabeled data; and reinforcement learning, where agents learn by interacting with an environment and receiving feedback in the form of rewards or penalties.

PART - B:

2. a) Explain about greedy best-first search technique.
1. Concept: Greedy best-first search is a search algorithm that selects the path that appears to be the most promising at each step. It makes decisions based on the immediate cost of the path and does not consider the history of the path.
2. Heuristic Function: It uses a heuristic function to estimate the cost to reach the goal from the current node. This function helps prioritize nodes closer to the goal.
3. Implementation: The algorithm maintains a priority queue where nodes are prioritized based on the heuristic function's values.
4. Optimism: It is optimistic in nature, always hoping to find a solution in the direction that looks most promising. This can lead to quick solutions in some cases.
5. Not Optimal: It does not guarantee an optimal solution. The path chosen might not be the best overall path to the goal.
6. Completeness: It is not complete, meaning it does not always find a solution if one exists, especially in infinite or very large search spaces.
7. Space and Time Efficiency: It can be more space and time-efficient than other algorithms like breadth-first search, especially in large search spaces.

8. Application: Commonly used in problems where a reasonable heuristic is available, such as pathfinding in maps or puzzle games.
9. Heuristic Accuracy: The effectiveness heavily depends on the accuracy of the heuristic function. A bad heuristic can lead to poor performance.
10. Real-World Examples: It's used in scenarios like routing protocols where immediate cost estimation is crucial, and the overall optimality is secondary.

2. b) What simulated annealing search? Explain.

1. Inspiration from Physics: Simulated annealing is inspired by the annealing process in metallurgy. It is a probabilistic technique used for finding an approximate global optimum of a given function.
2. Overcoming Local Optima: Unlike other methods that can get stuck in local optima, simulated annealing has the ability to escape local optima to increase the chance of finding the global optimum.
3. Temperature Concept: It incorporates a parameter called "temperature," which gradually decreases over time. High temperatures allow the algorithm to explore the search space more freely, accepting even bad moves.
4. Cooling Schedule: The rate at which the temperature decreases (cooling schedule) is crucial. Too fast cooling might lead to suboptimal solutions; too slow cooling increases computation time.
5. Randomness: It probabilistically decides whether to accept worse solutions during the search process, which decreases as the temperature lowers.
6. Optimization Problems: Ideal for optimization problems where the search space is large and the global optimum is difficult to find.
7. Trade-off Between Exploration and Exploitation: Initially, it focuses on exploration (searching broadly) and gradually shifts towards exploitation (searching locally around promising areas).
8. Flexibility: Can be applied to various kinds of optimization problems, including those with discrete or continuous variables.
9. Stopping Criterion: Typically, the algorithm stops when a certain temperature is reached or no improvement is observed over a number of iterations.
10. Applications: Widely used in fields like operations research, computer science, and artificial intelligence, particularly in scheduling, routing, and complex optimization tasks.

3. a) Discuss about A* search and Bidirectional search techniques.

A* Search:

1. Heuristic Involvement: A* search uses heuristics to guide its search for the shortest path, making it more efficient than uninformed search algorithms.
2. Cost Function: The algorithm utilizes a cost function, `f(n) = g(n) + h(n)`, where `g(n)` is the cost from the start node to `n`, and `h(n)` is the estimated cost from `n` to the goal.

3. Optimality: A* is optimal if the heuristic function `h(n)` is admissible, meaning it never overestimates the actual cost.
4. Completeness: The algorithm is complete, ensuring that it will find a solution if one exists.
5. Efficiency: While more efficient than other uninformed searches, its efficiency depends on the quality of the heuristic.
6. Best-First Search: A* is a form of best-first search, where nodes are selected for expansion based on their estimated total cost.
7. Memory Consumption: A* can be memory-intensive as it keeps all generated nodes in memory.
8. Dynamic Path Adjustment: A* can adjust its path dynamically if a better path is found during the search.
9. Application: Commonly used in pathfinding and graph traversal, particularly in maps and games.
10. Heuristic Design: The design of the heuristic function is critical and problem-specific, influencing the effectiveness and efficiency of the search.

Two-Way Search

1. Two-Way Search: It simultaneously searches forward from the start node and backward from the goal node, meeting in the middle.
2. Speed: Generally faster than traditional unidirectional search methods, as it effectively reduces the search space.
3. Memory Usage: Can be more memory efficient compared to A*, as it may explore fewer nodes in total.
4. Optimality: It is optimal if both searches are performed with an optimal algorithm like Breadth-First Search.
5. Completeness: Bidirectional search is complete if the search space is finite and both searches are complete.
6. Implementation Complexity: More complex to implement compared to unidirectional searches due to the need to manage two search fronts and identify the meeting point.
7. Suitable Problems: Most effective in problems with unique start and goal states, such as pathfinding in a maze.
8. Meeting Point Identification: One of the key challenges is efficiently identifying the meeting point of the two search fronts.
9. Limited Applicability: Not all problems are amenable to bidirectional search, especially those without a clearly defined goal state or reversible actions.
10. Heuristic Compatibility: Unlike A*, bidirectional search does not inherently utilize heuristics, although heuristic versions exist.

3.b) Explain in detail about online search agents.

Online search agents operate in an unknown environment and make decisions based only on the current percept. They do not have prior knowledge of the

environment and must learn and adapt as they operate. Here are key points about online search agents:
1. Percept-Based Decisions: They make decisions based on current perceptions without complete knowledge of the environment.
2. Environment Learning: Continuously learn about the environment as they explore.
3. Real-Time Operation: Designed to operate in real-time, making quick decisions without extensive deliberation.
4. Limited Lookahead: Typically use a limited lookahead strategy due to the real-time constraints and unknown environment.
5. Adaptability: Must be adaptable to new information and changes in the environment.
6. Exploration vs. Exploitation: They face the challenge of balancing the exploration of unknown areas with the exploitation of known beneficial paths.
7. Uncertainty Handling: Capable of handling uncertainties and dynamic changes in the environment.
8. State Space Representation: Often represent the state space incrementally as they discover it.
9. Goal-Oriented Behavior: While exploring, they aim to fulfill specific goals or tasks.
10. Heuristics and Learning Algorithms: May use heuristics or learning algorithms to improve decision-making over time.

4. a) What are knowledge-based agents? How can a knowledge-based agent be described at three levels?

Knowledge-Based Agents:
1. Definition: Knowledge-based agents are AI systems that make decisions and act based on the knowledge they possess.
2. Knowledge Representation: They use formalized knowledge representation to interpret, predict, and reason about the world.
3. Inference Mechanism: Employ an inference mechanism to draw new conclusions from existing knowledge.
4. Dynamic Knowledge Update: Capable of updating their knowledge base as they acquire new information.
5. Decision Making: Make decisions based on logical deductions from their knowledge base.
6. Expert Systems: Often used in expert systems for specific domains, like medical diagnosis or legal advice.
7. Rule-Based Reasoning: Typically use rule-based reasoning, although more advanced agents may use probabilistic models.
8. Goal-oriented: Act to achieve specific goals based on their knowledge and reasoning.

9. Interaction with Environment: Can interact with their environment and adapt their knowledge base accordingly.
10. Limitations: Their effectiveness is limited by the completeness and accuracy of their knowledge base.

Three Levels of Description for Knowledge-Based Agents:

1. Knowledge Level: Describes the agent's knowledge about the world, including facts, objects, and relations between them. It is abstract and focuses on what the agent knows, irrespective of how the knowledge is represented or used.
2. Logical Level: Details how the agent's knowledge is represented, often using logical formalisms like propositional or first-order logic. This level deals with the syntax and semantics of the knowledge representation.
3. Implementation Level: Describes how the agent's knowledge and reasoning are actually implemented, often involving data structures, algorithms, and programming languages. This level is concerned with the practical aspects of building the agent in a computing environment.

4. b) Give ontological and epistemological commitments of propositional logic.

Ontological Commitments of Propositional Logic

1. Abstract Entities: Propositional logic deals with abstract entities like propositions, which are either true or false.
2. Binary Truth Values: It commits to a world where statements have binary truth values – true or false, with no intermediate values.
3. Atomic Structure: The ontology assumes the existence of atomic propositions that cannot be further decomposed.
4. Existence of Logical Connectives: It recognizes the existence of logical connectives (AND, OR, NOT, etc.) as fundamental operations on propositions.
5. Non-Existence of Quantifiers: Unlike first-order logic, propositional logic does not commit to quantifiers (like "forall" or "exists"), thus not dealing with objects or their properties directly.
6. Formal Structure Emphasis: The focus is on the formal structure of arguments rather than the content of the propositions.
7. Discrete Framework: It operates within a discrete framework where propositions are distinct and separate.
8. Independence from Reality: The truth values of propositions are independent of their correspondence to real-world facts.
9. Simplicity of Entities: The ontology is minimalistic, considering only the simplest form of propositions.
10. No Commitment to Specific Entities: Propositional logic does not commit to any specific entities in the world beyond the abstract propositions and their relations.

Epistemological Commitments of Propositional Logic

1. Knowledge as Logical Structure: Knowledge is understood in terms of logical relations and structures among propositions.
2. Objective Truth: It assumes an objective notion of truth that is independent of human beliefs or knowledge.
3. Deductive Reasoning: The emphasis is on deductive reasoning, deriving conclusions from given premises through logical inference.
4. Formal Proof and Verification: Knowledge validation is through formal proof and logical verification.
5. Binary Knowledge Assessment: Knowledge claims are assessed in binary terms – a statement is either known (true) or not known (false).
6. Syntactical Knowledge Representation: Knowledge is represented syntactically, without regard to semantic content.
7. Rule-Based Understanding: Understanding is based on the application of fixed logical rules.
8. Static Knowledge Representation: Propositional logic represents knowledge in a static manner, without considering changes over time.
9. Absence of Contextual Interpretation: Knowledge is interpreted without context, focusing solely on logical form.
10. Universal Applicability: It assumes the universal applicability of logical principles to any propositional content.

5. a) Define the constraint satisfaction problem and explain the backtracking search for CSPs.

1. Basic Definition: A Constraint Satisfaction Problem (CSP) is a mathematical question defined as a set of objects whose state must satisfy a number of constraints or limitations.
2. Components: CSPs are composed of three main components: variables, domains, and constraints. Variables are the unknowns to be solved, domains are the set of possible values that each variable can take, and constraints are the rules that limit the values the variables can simultaneously take.
3. Applications: CSPs are used in various fields such as artificial intelligence, computer science, and operations research. Common applications include scheduling, planning, resource allocation, and puzzle solving.
4. Formulation: In a CSP, one aims to assign values to each variable so that all constraints are satisfied. The solution to a CSP is the assignment of values to all variables that satisfy all constraints.
5. Complexity: CSPs can range from being simple and easily solvable to being highly complex and NP-complete, depending on the number and nature of the constraints and the size of the domains.
6. Consistency Checking: A key part of solving CSPs is checking for consistency, ensuring that constraints are not violated as values are assigned to variables.

7. Global Constraints: Some CSPs involve global constraints that affect multiple variables and are more complex than constraints affecting only a pair of variables.
8. Representation: CSPs are often represented graphically, where variables are nodes and constraints are edges connecting the nodes.
9. Optimization Version: In some cases, CSPs are extended to optimization problems, where the goal is to find the best solution according to a certain criterion while still satisfying the constraints.
10. Variability in Constraints: Constraints in a CSP can be either hard constraints, which must be strictly met, or soft constraints, which are preferable but not mandatory.

Backtracking Search for CSPs

1. Basic Concept: Backtracking search is a depth-first search algorithm for solving CSPs. It incrementally builds candidates to the solutions and abandons a candidate as soon as it determines that the candidate cannot possibly be completed to a valid solution.
2. Variable Assignment: The algorithm starts by selecting a variable and assigning it a value from its domain.
3. Forward Checking: After each assignment, forward checking is often used to reduce the domains of the remaining variables by removing values that violate constraints.
4. Backtracking: If the algorithm reaches a point where a constraint is violated, it backtracks, undoing the most recent variable assignment and tries a different value.
5. Order of Variable Selection: The efficiency of the backtracking search can significantly depend on the order in which variables are selected. Heuristics like Minimum Remaining Values (MRV) can be used to choose variables.
6. Value Ordering: Similarly, the order in which values are assigned to variables matters. Heuristics like the Least Constraining Value can be used to choose values that impose the fewest restrictions on other variables.
7. Pruning: The search space can be pruned using techniques like constraint propagation, where the constraints are used to further reduce the domains of variables.
8. Completeness: The backtracking algorithm is complete, meaning it will find a solution if one exists, or conclude that no solution exists.
9. Efficiency: While backtracking is a powerful method, it can be inefficient for large, complex CSPs due to its brute-force nature.
10. Improvements: Various improvements to basic backtracking, like backjumping, constraint learning, and dynamic variable ordering, can help overcome inefficiencies and solve more complex CSPs more effectively.

5. b) What is Adversarial Search? Explain adversarial search techniques.

Adversarial search is a type of search in artificial intelligence and game theory that deals with decision-making in environments where players compete against each other. The goal is to find the best move in a situation where an opponent is trying to minimize your chances of winning. This search is fundamental in games like chess, checkers, or Go, where two players take turns making moves.

Adversarial Search Techniques:

1. Minimax Algorithm:

A classic technique used in two-player zero-sum games.

Players are categorized as Minimizer and Maximizer.

The algorithm explores the game tree by considering all possible moves, their consequences, counter-moves, and so on.

It aims to minimize the possible loss for a worst-case scenario.

2. Alpha-Beta Pruning:

An optimization of the minimax algorithm.

It prunes (cuts off) branches in the game tree that need not be searched because there already exists a better move available.

It reduces the number of nodes evaluated in the game tree, thus speeding up decision-making.

3. Heuristic Evaluation Functions:

Used in games with a large search space.

These functions evaluate the desirability of a game position when it's not feasible to search the game tree until the end.

They provide a way to estimate the advantage or disadvantages in a given position.

4. Negamax Search:

A variation of minimax that simplifies implementation by negating the values returned from recursive calls.

Assumes that the players have exactly opposite utility values.

5. Iterative Deepening:

Used in conjunction with other search techniques.

Involves repeatedly deepening the limit of a depth-first search, combining the benefits of depth-first and breadth-first searches.

6. Monte Carlo Tree Search (MCTS):

Uses randomness to explore the search space.

It balances between exploring unexplored moves and exploiting known good moves.

Particularly popular in games like Go.

7. Transposition Tables:

The store previously explored positions to avoid redundant calculations.

Useful in games where the same position can be reached by different sequences of moves.

8. Zobrist Hashing:

A technique to efficiently store game board positions in a transposition table.

Reduces the memory footprint and speeds up the retrieval of stored positions.
9. Principal Variation Search (PVS):
A refinement of alpha-beta pruning.
Tries to improve the efficiency of alpha-beta pruning by searching the likely best move first.
10. Opening Books and Endgame Databases:
Pre-calculated best moves in the opening and endgame phases of a game.

6. a) Write about resolution in first-order logic.
1. Definition: Resolution in First-Order Logic (FOL) is a fundamental rule of inference used for deductive reasoning and automated theorem proving.
2. Extension of Propositional Logic: It extends the resolution principle from propositional logic to handle quantifiers and relations, which are key components of FOL.
3. Clauses and Literals: Involves the use of clauses (disjunctions of literals) where literals are atomic formulas or their negations.
4. Unification Process: Central to resolution in FOL, unification involves finding a common instance for different literals by appropriately substituting variables.
5. Rule of Resolution: If two clauses contain complementary literals (a literal and its negation), they can be resolved by forming a new clause with the remaining literals.
6. Refutation Technique: Often used to prove the falsity of a statement by showing that its negation leads to a contradiction.
7. Iterative Process: Repeated application of the resolution rule can derive new clauses, progressively simplifying the problem.
8. Completeness: Resolution is a complete inference rule in FOL, meaning it can derive any logical consequence from a set of premises.
9. Automated Theorem Proving: Widely used in AI for automated theorem proving, where conclusions are systematically derived from premises.
10. Complexity and Efficiency: While powerful, resolution in FOL can be complex and computationally intensive, often requiring optimization techniques in practical applications.

6. b) What is Ontological engineering? Explain.
1. Definition: Ontological Engineering involves the practice and methodology of creating, maintaining, and applying ontologies.
2. Ontologies: These are formal representations of knowledge with a rich structure of concepts and relationships pertinent to a specific domain.
3. Concepts and Relationships: Involves defining and structuring key concepts, entities, and their interrelations within a domain.
4. Standardization of Knowledge: Aims to create a common framework and language for representing domain-specific knowledge.

5. Semantic Web: Plays a critical role in the development of the Semantic Web, enabling machines to understand and process web content.
6. Knowledge Sharing and Reuse: Facilitates the sharing and reuse of domain knowledge across various systems and communities.
7. AI and Machine Learning: Provides structured knowledge that can enhance AI and machine learning applications.
8. Interoperability: Ensures that different systems and organizations can effectively communicate and exchange information.
9. Domain Analysis and Modeling: Involves rigorous analysis and modeling of the domain to ensure accurate representation.
10. Dynamic and Evolving: Ontologies are not static; they evolve as new knowledge is acquired and as the domain changes.

7.a) Explain the semantics of first-order logic in knowledge representation.
1. Basic Elements: Includes objects, functions, and relations, along with constants, variables, and quantifiers.
2. Interpretation of Symbols: Each symbol (like constants, and predicates) is assigned a specific meaning or reference in the domain.
3. Assignment of Truth Values: Semantics determine how truth values are assigned to sentences based on their interpretation.
4. Models: A model is a specific assignment of meanings that makes a set of sentences true.
5. Quantifiers: 'Existential' and 'Universal' quantifiers add depth to the logic, allowing for more complex statements about the world.
6. Logical Connectives: Include and, or, not, implies, and if-and-only-if, used to build complex sentences from simpler ones.
7. Domain of Discourse: Refers to the collection of objects being discussed and is crucial for interpreting variables and quantifiers.
8. Validity and Satisfiability: A formula is valid if true in all models, and satisfiable if true in at least one model.
9. Role in Reasoning: Semantics are essential for deducing new knowledge and verifying information logically.
10. Foundation for Advanced AI: Provides a rigorous foundation for advanced AI applications in knowledge representation and reasoning.

7. b) Explain about backward chaining in first-order logic.
1. Goal-Driven Approach: Starts with the goal or conclusion and works backward to find supporting evidence.
2. Use in Expert Systems: Commonly used in rule-based expert systems for diagnostic and problem-solving tasks.
3. Recursive Process: Involves recursively proving the premises of rules that could lead to the goal.

4. Hypothesis Testing: Essentially tests hypotheses by looking for evidence that supports or refutes them.
5. Rule Selection: Select rules that directly relate to the current goal or subgoal.
6. Efficiency: Often more efficient than forward chaining when the number of potential conclusions is large.
7. Depth-First Search: Typically employs a depth-first search strategy through the rule base.
8. Data-Driven: Requires a base of known facts or data from which to reason backward.
9. Termination Conditions: Continues until it either proves the initial goal, proves it is false, or exhausts all rule paths.
10. Applications: Widely used in areas like medical diagnosis, technical support, and legal reasoning.

8. List and explain different classical planning approaches.

1. State-Space Planning: Involves searching through a space of possible states of the world. Plans are sequences of actions transforming the initial state into a goal state. This approach uses graph search algorithms like A* or Dijkstra's algorithm.
2. Plan-Space Planning: Instead of searching through states, this approach searches through the space of partial plans. Algorithms iteratively refine a partial plan until it meets the goal criteria, handling one sub-goal at a time.
3. Hierarchical Task Network (HTN) Planning: Break down tasks into subtasks, creating a hierarchy. It starts with a high-level task and progressively decomposes it into more specific and smaller tasks, which are easier to manage and solve.
4. Satisfiability Planning (SATPLAN): Transforms the planning problem into a satisfiability problem. The idea is to encode the planning problem into a Boolean formula and then use a SAT solver to find a solution.
5. Strips Planning: Named after the Stanford Research Institute Problem Solver, this is one of the earliest planning methods. It uses a simple representation of actions in terms of preconditions and effects.
6. Graphplan: Builds a graph that captures the relationships between actions and propositions over time. It then searches this graph for a valid plan. It's efficient due to its compact representation of actions and states.
7. Partial-Order Planning (POP): Plans are sets of actions with constraints on their ordering, allowing actions to be ordered only as necessary. This approach supports more flexibility and parallelism in plans.
8. Case-Based Planning: Involves reusing plans from previous similar situations. It adapts solutions from past problems to new problems, reducing the need for extensive search.

9. Reactive Planning: Focuses on achieving goals in dynamic, unpredictable environments. It uses a set of condition-action rules to determine the best action to take based on the current state.
10. Constraint-Based Planning**: Uses constraints to express the relationships between different parts of the plan. The planner tries to find a set of actions that satisfies all the constraints.

9. a) Explain forward state space search with an example.
1. Definition: Forward state space search is a problem-solving technique used in artificial intelligence, where the solution is found by starting from an initial state and applying actions to reach a goal state.
2. Graph Representation: The state space is often represented as a graph, where nodes represent states and edges represent the actions that transform one state into another.
3. Breadth-First and Depth-First: Common strategies for traversing this space include breadth-first search (BFS) and depth-first search (DFS).
4. Example - Puzzle Solving: Consider solving a simple 3x3 sliding puzzle. The initial state is the mixed puzzle, and the goal state is the puzzle arranged in a particular order.
5. Applying Actions: In each step, you slide a tile, creating a new state. Each move represents a transition in the state space.
6. Exploring States: The algorithm explores various sequences of moves (states) to find a sequence that leads to the goal state.
7. Optimality: BFS guarantees finding the shortest path to the goal, while DFS uses less memory but may not find the shortest path.
8. Complexity: The main challenge is the size of the state space, which can grow exponentially with the complexity of the problem.
9. Heuristics: Heuristic functions can be used to guide the search more efficiently towards the goal state.
10. Applications: Widely used in games, puzzle solving, robot navigation, and other AI applications where a sequence of actions leads to a desired outcome.

9. b) Discuss about hierarchical planning with an example.
1. Definition: Hierarchical planning involves breaking down a complex problem into smaller, more manageable sub-problems or tasks.
2. Hierarchical Task Network (HTN): This is a common approach where tasks are decomposed into subtasks in a hierarchy.
3. Top-Down Approach: It starts with high-level tasks and progressively breaks them down into detailed actions.
4. Example - Travel Planning: Consider planning a trip from home to a foreign country.

5. High-Level Planning: The high-level task is to plan the trip. This can be broken down into booking flights, arranging accommodations, and planning local travel.
6. Further Decomposition: Each of these tasks can be further decomposed. For example, booking flights involves choosing dates, selecting flights, and making payments.
7. Dependencies and Sequencing: The planner must consider task dependencies (e.g., booking accommodations after flights) and sequence tasks appropriately.
8. Reusability: Subtasks in hierarchical planning can often be reused in different contexts or for different problems.
9. Efficiency: This approach is more efficient than flat planning as it reduces the complexity by focusing on smaller parts of the problem at a time.
10. Applications: Hierarchical planning is used in a wide range of applications, from robotics (where a robot might have tasks like navigating, picking up objects, etc.) to complex project management in software development.

10.a) Discuss in brief about Dempster-Shafer theory.

1. Basic Concept: Dempster-Shafer theory, also known as the theory of belief functions, is a mathematical theory of evidence. It allows one to combine evidence from different sources and arrive at a degree of belief (or probability) that takes into account all the available evidence.
2. Belief Functions: It works with belief functions, which represent the probability of an event based on the available evidence, differing from traditional probability by not requiring mutually exclusive and exhaustive hypotheses.
3. Belief and Plausibility: The theory provides two measures: belief and plausibility, which together define a range of probability, reflecting the uncertainty due to incomplete information.
4. Evidence Combination: A key feature is the combination rule, also known as Dempster's rule of combination, which allows for the aggregation of independent pieces of evidence.
5. Handling Uncertainty: It's particularly useful in situations where information is incomplete or uncertain, as it provides a framework for modeling and reasoning with uncertainty.
6. Frame of Discernment: The theory is based on a frame of discernment, which is a set of all possible outcomes. The power set of this frame forms the basis for defining belief functions.
7. Flexibility Over Probability Theory: Dempster-Shafer theory is more flexible than traditional probability theory, as it does not require precise probabilities for each event.
8. Non-specificity: The theory allows for non-specific evidence, meaning that evidence can support sets of hypotheses rather than specific ones.

9. Application Areas: It is widely used in areas such as artificial intelligence, decision-making, and statistical analysis, where uncertainty is a key factor.
10. Criticism and Challenges: Despite its strengths, the theory faces criticism, particularly regarding the computational complexity and interpretation of the combination rule, especially when dealing with conflicting evidence.

10. b) How to represent knowledge in an uncertain domain? Explain

1. Probabilistic Models: These models assign probabilities to different outcomes or hypotheses, accounting for uncertainty in knowledge representation. Bayesian networks are a common probabilistic model.
2. Fuzzy Logic: Unlike classical logic, fuzzy logic deals with reasoning that is approximate rather than fixed and exact. It is particularly useful in handling the concept of partial truth.
3. Rule-Based Systems with Uncertainty Factors: In such systems, each rule is associated with an uncertainty factor, indicating the likelihood of the rule being correct.
4. Belief Networks: Also known as Bayesian networks, these graphical models represent variables and their conditional dependencies via a directed acyclic graph.
5. Dempster-Shafer Theory: As mentioned earlier, this theory provides a mechanism for combining evidence to calculate degrees of belief.
6. Neural Networks: These computational models can represent complex patterns and uncertainties, learning from examples and making decisions based on probabilities.
7. Case-Based Reasoning: This method involves solving new problems based on the solutions of similar past problems, allowing for uncertainty in matching cases.
8. Default Reasoning: This approach involves reasoning with default rules that hold in general but may have exceptions, thus handling uncertainty in rule applicability.
9. Ontologies with Uncertainty: Ontologies can be extended to represent uncertain information, defining concepts, relationships, and properties in a way that accommodates ambiguity and vagueness.
10. Stochastic Methods: These methods involve the use of random variables and processes to model and reason about uncertainty, particularly in dynamic systems.

11. a) Explain Bayes' rule and its uses.

1. Definition: Bayes' Rule, also known as Bayes' Theorem, is a fundamental principle in probability theory. It describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

2. Formula: The theorem is mathematically expressed as P(A|B) = [P(B|A) * P(A)] / P(B), where P(A|B) is the probability of A given B, P(B|A) is the probability of B given A, P(A) and P(B) are the probabilities of A and B independently.
3. Updating Beliefs: Bayes' Rule is used for updating the probability estimate for a hypothesis as more evidence or information becomes available.
4. Medical Diagnosis: In healthcare, it aids in diagnosing diseases based on the likelihood of a disease given a specific symptom or test result.
5. Machine Learning: It's fundamental in many machine learning algorithms, especially in Bayesian networks and probabilistic classifiers like Naive Bayes.
6. Spam Filtering: Employed in email spam filtering, Bayes' Rule helps determine the probability that an email is spam based on the presence of certain words.
7. Decision Making: In business and economics, it aids in making more informed decisions under uncertainty by updating the likelihood of different outcomes as new data is received.
8. Scientific Research: Used in hypothesis testing in scientific research, to update the probability of a hypothesis being true following experimental results.
9. Risk Assessment: In finance and insurance, it's used for risk assessment and actuarial analysis by updating the probabilities of various risks based on new information.
10. Natural Language Processing (NLP): Used in NLP for tasks such as text classification and sentiment analysis by updating probabilities based on the occurrence of certain words or phrases.

11. b) Discuss in brief about explanation-based learning.

1. Concept: Explanation-based learning is a form of machine learning that focuses on understanding and analyzing the reasoning behind the conclusions drawn.
2. Generating Explanations: EBL involves generating explanations for the decisions or classifications made by a learning system, based on its understanding of the domain.
3. Improving Performance: The goal is to improve the performance of a learning system by focusing on the relevant aspects of the input data, using explanations to guide the learning process.
4. Domain-Specific Knowledge: EBL heavily relies on domain-specific knowledge for generating explanations, making it more effective in domains where such knowledge is available.
5. Learning from Few Examples: It can learn effectively from a small number of examples by understanding the underlying principles, unlike other methods that require large datasets.
6. Rule Refinement: EBL can refine existing rules or knowledge in a system by analyzing exceptions or failures and understanding their explanations.

7. Focus on Causality: It emphasizes understanding the causal relationships in the data, which can lead to more robust and interpretable models.
8. Generalization: EBL can generalize from specific instances to broader rules or concepts by understanding the underlying explanations.
9. Applications: It's used in areas such as robotics, where understanding the reasoning behind actions is crucial and in complex problem-solving tasks.
10. Limitations: The effectiveness of EBL depends heavily on the quality and completeness of the domain knowledge it uses, limiting its applicability in domains where such knowledge is incomplete or unavailable.