Reg.No _____

SNS COLLEGE OF TECHNOLOGY

(Autonomous)

MCA- Internal Assessment –II (Mar 2024)

A

Academic Year 2023-2024(EVEN) / Sixth Semester

19CSE303 – Artificial Intelligence

Time: 1 1/2 Hours **Maximum Marks: 50**

Answer All Questions

PART - A (5 x 2 = 10 Marks)

		CO	
1	**What is the most general unifier of the terms p(x, y) and p(a, b)?** General unifier = { x/a, y/b }	CO2	Und
2	**In the context of forward chaining, what is a lift?** A lift in forward chaining is the process of adding new inferred facts or rules to the agenda for further evaluation.	CO2	Ana
3	**How does the partial-order planning algorithm resolve threats during the planning process?** The partial-order planning algorithm resolves threats by either introducing ordering constraints between actions or by adding additional actions to the plan to address the conflicts.	CO3	Ana
4	**Define what is meant by a "threat" in the context of partial-order planning.** In partial-order planning, a "threat" refers to a potential conflict where the effects of one action could undo or interfere with the preconditions of another action in the plan.	CO3	Und
5	**Describe the two types of nodes that alternate in levels of a planning graph and their purpose.** 1. Literal Level Node **2.** Action Level Nodes	CO3	Rem

PART - B (2 x 13 = 26 Marks, 1 X 14 = 14 Marks)

6 (a) **Starting with the following knowledge base and query, show the steps of forward chaining to derive the goal: KB: p(a), q(x) ∧ r(x) → s(x), s(x) ∧ t(x) → u(x)**

1. Initial State:

Known facts:
p(a) (from the KB)

2. Checking Rules:

We have two rules in the KB:

Rule 1: q(x) ∧ r(x) → s(x) CO2 Ana
Rule 2: s(x) ∧ t(x) → u(x)
Since we only have p(a) currently, neither rule can be applied yet.

3. **No Match, Continue Checking:**

We haven't derived anything new, so we keep checking the rules.

4. **No New Facts, Termination:**

There are no facts in the KB that match the variables in the first parts (premises) of the rules.

(Or)

(b) **Analysis the importance of factoring and subsumption in resolution-based theorem proving, and how they can improve efficiency.** CO2 Ana

Factoring identifies and removes redundant information within a single clause. Imagine a clause like "A ∨ B ∨ C ∨ (A ∧ B)". The implication of "(A ∧ B)" is already captured by "A ∨ B". Factoring eliminates this redundancy, resulting in a simpler clause like "A ∨ B ∨ C".

Subsumption identifies a clause that is logically implied by another clause. For instance, the clause "A ∨ B" subsumes "A ∨ B ∨ C" because any situation satisfying "A ∨ B" also satisfies "A ∨ B ∨ C". Subsumption allows us to discard the subsumed clause, focusing on the more general one.

These techniques improve efficiency in resolution-based theorem proving by:

- **Reducing Search Space:** By eliminating redundant information and subsumed clauses, the overall number of clauses to be considered during the proof search process is reduced. This leads to faster exploration of the search space and potentially quicker discovery of a proof (if one exists).
- **Focus on Relevant Information:** Factoring ensures that clauses only contain essential information, while subsumption directs the reasoning process towards more general clauses with broader implications. This helps the theorem prover concentrate on the most relevant aspects of the problem.

7 (a) **Propose a strategy for continuous planning and monitoring to handle unexpected events for an autonomous robot operating in an office mail delivery scenario.** **CO2** **App**

Planning:

1. **High-Level Planning:** Create a master plan for the entire delivery route before deployment. This includes:
    - **Map Integration:** Load a detailed map of the office, including designated delivery points and potential obstacles.
    - **Delivery Schedule:** Integrate delivery schedules with the master plan, optimizing route efficiency.
    - **Contingency Plans:** Pre-program basic responses for common unexpected events. These could include:
        - Doorways blocked - Find alternate routes or request assistance.
        - Low battery - Navigate to the nearest charging station.
        - Object in path - Replan route or carefully maneuver around the obstacle.
2. **Dynamic Replanning:** Implement a dynamic replanning module that constantly monitors sensor data and updates the plan in real-time. This includes:
    - **Sensor Fusion:** Combine data from LiDAR, cameras, and other sensors to build a comprehensive picture of the environment.
    - **Real-time Path Adjustment:** Use sensor data to identify and avoid obstacles not present in the initial plan.
    - **Priority Management:** Prioritize tasks based on urgency (e.g., critical deliveries) and adapt the plan accordingly.

Monitoring:

1. **Sensor Data Analysis:** Continuously analyze sensor data for anomalies that might indicate unexpected events. This includes:
    - **Object Detection:** Identify and classify unexpected objects in the

robot's path.
    - **Movement Detection:** Detect people or moving objects that could cause collisions.
    - **Environmental Changes:** Monitor for changes in lighting, temperature, or other environmental factors that could impact navigation.
2. **Decision Making:** Based on sensor data analysis, the robot should be able to make informed decisions:
    - **Continue with Plan:** If no anomalies are detected, continue with the current plan.
    - **Trigger Replanning:** If an unexpected event is detected, trigger a replanning process to adapt the route or task.
    - **Request Assistance:** For complex situations beyond its capabilities, request assistance from a human operator.
3. **Learning and Adaptation:** Continuously update the robot's knowledge base with encountered situations and successful responses. This allows the robot to improve its ability to handle unexpected events in the future.

(Or)

(b) **For a simple logistics planning problem, construct the first three levels of the planning graph given the initial state, actions, and goal.**

Initial State:

- Truck is at the Warehouse (W).
- Package A and Package B are both at the Warehouse (W).

Actions:

- Pickup Package A (W, A)
- Pickup Package B (W, B)
- Deliver Package A (A)
- Deliver Package B (B)

Goal:

- Truck is at any location (Doesn't matter where as long as deliveries are done). **CO3** **App**
- Package A is delivered (Delivered to Customer A).
- Package B is delivered (Delivered to Customer B).

Planning Graph (First 3 Levels):

Level 0 (Initial State):

State: W (Truck at Warehouse, Packages A & B at Warehouse)

Level 1 (Actions applicable to Initial State):

W -> (Pickup Package A) -> A1
W -> (Pickup Package B) -> B1

Level 2 (Possible outcomes of Level 1 actions):

State A1 (Result of Pickup Package A):

- Truck: A (Truck now has Package A)
- Package A: In Transit (Picked up from Warehouse)
- Package B: W (Still at Warehouse)

State B1 (Result of Pickup Package B):

- Truck: B (Truck now has Package B)
- Package A: W (Still at Warehouse)
- Package B: In Transit (Picked up from Warehouse)

Explanation:

- Level 0 represents the initial state of the problem.
- Level 1 shows the possible actions (picking up packages) that can be taken from the initial state.
- Level 2 shows the resulting states after each action is applied. In this case, picking up Package A leads to state A1 and picking up Package B leads to state B1.

8 (a) **Suppose the route planning is formulated as a state-space search problem. What would be an appropriate search algorithm (e.g. breadth-first, depth-first, A*, etc.) and heuristic function for finding efficient drone delivery routes? Justify your choice.**

- **Breadth-First Search (BFS):** While BFS guarantees finding an optimal solution, it explores all states at a given depth before moving to the next. This can be inefficient for route planning as BFS might explore many irrelevant paths before finding the optimal one, especially for large delivery areas.
- **Depth-First Search (DFS):** DFS explores a single path deeply until it reaches a goal state or a dead end. This can be problematic for drone deliveries as DFS might get stuck in a long, inefficient route while potentially shorter paths exist unexplored.
- *A Search:** A* search combines the benefits of BFS and DFS. It uses a heuristic function to estimate the cost (distance or time) of reaching the goal state from any state. A* prioritizes exploring states with a lower total cost (actual travel cost + heuristic estimate). This makes A* efficient as it focuses on promising routes while guaranteeing an optimal solution like BFS.

CO3 **Eva**

Heuristic Function for Drone Delivery Routes

An appropriate heuristic function (h(n)) for drone delivery routes should estimate the remaining cost to reach the final destination from a current state (n). Here are two options:

1. **Straight-Line Distance Heuristic:** This calculates the straight-line distance between the current delivery location and the final destination. This is a simple and fast heuristic but might overestimate the actual travel cost due to obstacles or no-fly zones.
2. **Nearest Neighbor Heuristic:** This estimates the cost by calculating the sum of straight-line distances between remaining delivery locations, considering the current location and the final destination as part of the remaining deliveries. This provides a more realistic estimate than the straight-line approach but might not account for the most efficient order of deliveries.

(Or)

(b) **Imagine you have a robotic arm designed to make coffee. Specify the preconditions and effects of each action (e.g., precondition for turn on kettle is the kettle is empty).**

Grinding Beans:

- **Precondition:**
  - Coffee beans are loaded in the bean hopper.
  - Grinder is properly assembled and plugged in.
- **Effect:**
  - A specific amount of coffee grounds are dispensed into the portafilter (or filter basket).

Filling Portafilter:

- **Precondition:**
  - Portafilter is attached to the grinder.
  - Coffee grounds bin is empty (or ready to receive grounds).
- **Effect:**
  - Portafilter is filled with the desired amount of coffee grounds.

Tamping Coffee:

- **Precondition:**
  - Portafilter is filled with coffee grounds.
  - Tamper is attached to the robotic arm.
- **Effect:**
  - Coffee grounds are evenly tamped in the portafilter for optimal extraction.

Positioning Portafilter:

- **Precondition:**
  - Portafilter is filled and tamped with coffee grounds.
  - Espresso machine is on and preheated.
- **Effect:**
  - Portafilter is locked into the espresso machine's group head.

CO3 **Eva**

Brewing Coffee:

- **Precondition:**
  - Portafilter is positioned in the espresso machine.
  - Water reservoir is filled.
- **Effect:**
  - Hot water is forced through the coffee grounds, extracting espresso into a cup placed beneath the portafilter.

Steaming Milk:

- **Precondition:**
  - Milk container is filled with cold milk.
  - Steam wand is attached to the robotic arm.
- **Effect:**
  - Milk is frothed and heated using the steam wand.

Cleaning:

- **Precondition:**
  - Coffee making process is complete.
  - Cleaning solution is available.
- **Effect:**
  - Portafilter, steam wand, and drip tray are cleaned to remove coffee residue.