Chapter IX - Classical Planning

9.1 Definition of Classical Planning

Classical Planning involves devising a sequence of actions to transition from an initial state to a goal state in a well-defined environment. The key characteristics of classical planning are:

1. **Deterministic Environment:** Actions have known and predictable outcomes.
2. **Discrete States:** The environment is modeled using a set of discrete states.
3. **Well-defined Actions:** Actions are predefined, and their effects are deterministic.
4. **Goal State:** The planner seeks to reach a specific goal state from an initial state.

The goal of classical planning is to find a **plan**—a sequence of actions that will take the system from an initial state to a goal state. Classical planners do not need to account for uncertainty or partial observability, making them simpler but less general compared to more advanced planning methods.

9.2 Planning Algorithms as State Space Search

In classical planning, **planning algorithms** often view the planning process as a **state space search**, where:

- **States** represent the configuration of the world at any point in time.
- **Actions** represent transitions between states.
- The objective is to find a sequence of actions (a plan) that leads from the **initial state** to the **goal state**.

Key Concepts:

1. **State Space Representation:**
   - The state space is typically represented as a **graph** where each node is a state, and each edge is an action that transitions between states.
   - The planner searches through the state space to find a path from the initial state to the goal state.
2. **Search Algorithms:**
   - **Uninformed Search:** Algorithms like **Breadth-First Search (BFS)** or **Depth-First Search (DFS)** can be used to explore the state space.

- **Informed Search:** Heuristic search algorithms, such as **A* search**, can guide the search by estimating the cost of reaching the goal.

Example:

A robot needs to plan a series of movements to navigate a maze from a start position to an end position. The maze is modeled as a state space, and the robot's movements are the actions that transition from one state to another.

9.3 Planning Graphs

A **Planning Graph** is a data structure used to represent a planning problem over time. It is a layered graph where each layer corresponds to a state of the system at a particular time step.

- **Nodes in the Planning Graph** represent either **propositions** (facts about the world) or **actions** (actions that can be executed).
- **Edges** in the graph connect actions to the propositions that they affect, and propositions to actions that can be used to achieve them.

Key Features:

1. **Levels of the Graph:**
   - Each level represents either a set of propositions (states) or actions.
   - **Proposition Level (P):** Contains the facts that hold at that moment in time.
   - **Action Level (A):** Contains the actions that can be taken to achieve certain propositions.
2. **Relaxed Planning:**
   - A **relaxed plan** ignores the delete effects of actions (assuming actions do not negate previous actions), making it easier to find a valid sequence of actions.
3. **Use in Planning:**
   - **GraphPlan Algorithm:** One of the most famous algorithms that uses planning graphs to find a valid plan. It constructs the graph iteratively and searches for a solution.

Example:

Imagine you want to make a sandwich. The **Propositions** would include "Bread on Table," "Peanut Butter on Bread," and "Sandwich Ready." The **Actions** would include "Pick up bread," "Spread peanut butter," and "Place peanut butter on bread." The planning graph helps in structuring and organizing these steps.

9.4 Other Classical Planning Approaches

Besides state-space search and planning graphs, there are other **classical planning techniques**, including:

1. **STRIPS (Stanford Research Institute Problem Solver):**
   - A formal language and method for representing actions and plans. Actions are represented with three parts: **preconditions** (what must be true before the action), **effects** (what changes after the action), and **delete lists** (which facts are negated).
   - STRIPS-based planners focus on finding sequences of actions to transition from the initial state to the goal.
2. **Partial Order Planning:**
   - In **partial order planning**, actions are not required to be ordered linearly. Instead, the planner defines a set of constraints on actions that must be respected, but the order in which actions occur is not strictly determined unless necessary.
   - It focuses on ordering the actions only when required, leading to potentially more efficient plans.
3. **Hierarchical Task Network (HTN) Planning:**
   - HTN planning decomposes high-level tasks into smaller sub-tasks and plans at different levels of abstraction. It is suitable for tasks with a natural hierarchical structure.
   - HTNs are commonly used in complex domains like robotics, where a broad task (e.g., "clean the house") is broken down into more specific actions (e.g., "vacuum living room," "wash dishes").

9.5 Analysis of Planning Approaches

Classical planning approaches differ in their trade-offs regarding **completeness**, **optimality**, **complexity**, and **flexibility**. Here's an analysis of the strengths and weaknesses of some key planning techniques:

1. **State-Space Search:**
   - **Advantages:** Simple to understand, can handle a wide variety of problems, and works well with small state spaces.
   - **Disadvantages:** Can become computationally expensive as the state space grows, leading to performance issues in large or complex domains.
2. **Planning Graphs:**

- **Advantages:** Efficient representation of planning problems, helps detect inconsistencies early, and supports relaxed planning.
- **Disadvantages:** Can be memory-intensive and may not always lead to the most efficient plan.

3. **STRIPS:**
   - **Advantages:** Well-established and widely used; simple to implement and provides a formal framework for action representations.
   - **Disadvantages:** Limited to deterministic environments and struggles with complex or partially observable environments.
4. **Partial Order Planning:**
   - **Advantages:** Flexible; avoids unnecessary ordering of actions and can potentially find shorter plans.
   - **Disadvantages:** May struggle with identifying interdependencies between actions in large, complex problems.
5. **HTN Planning:**
   - **Advantages:** Well-suited for tasks with clear hierarchical structures and allows more abstract reasoning.
   - **Disadvantages:** Can be difficult to formalize hierarchical tasks in complex domains; less suited for problems without inherent hierarchical structures.

Exercises

1. **State-Space Search Exercise:**
   Imagine a robot that needs to navigate from point A to point B in a grid. Define the initial state, the goal state, and describe the sequence of actions using state-space search. How would the robot use BFS or DFS to reach the goal?
2. **Planning Graph Exercise:**
   Create a planning graph for a simple cooking recipe (e.g., making a cup of tea). Identify the propositions, actions, and their interdependencies.
3. **STRIPS Exercise:**
   Define a STRIPS action schema for a robot to move from one room to another. Include the preconditions, effects, and delete lists for the action "Move" and any other relevant actions.
4. **Partial Order Planning Exercise:**
   Given a set of tasks to organize a party (e.g., "send invitations," "buy cake," "decorate room," "buy drinks"), create a partial order plan for these tasks. How do you determine which tasks must occur before others, and which tasks can be done in any order?

5. **HTN Planning Exercise:**
   Use HTN planning to break down the task "Make Dinner" into smaller tasks. What sub-tasks would you define, and how would you order them to make the process efficient?

This chapter introduced key **classical planning techniques** like **state-space search**, **planning graphs**, **STRIPS**, **partial order planning**, and **HTN planning**. Each technique has its strengths and trade-offs, and the choice of technique depends on the specific planning problem at hand. These techniques are foundational for building AI systems capable of solving complex tasks by reasoning about actions and their effects.