MODULE 6

➢ Planning in AI

Introduction:

- Planning in artificial intelligence involves decision-making processes executed by robots or computer programs to achieve predefined goals.
- It is a critical aspect of AI projects, essential for tasks ranging from route optimization to complex problem-solving.

1. **Forward State Space Planning (FSSP):**
    - FSSP operates similarly to forward state-space search.
    - It begins with an *initial state and progresses through a sequence of actions to reach the target state*.
    - Each action is applied to the current state, leading to a new state, until the goal state is achieved.
    - **Advantage:** Sound algorithm ensuring correctness.
    - **Disadvantage:** Large branching factor, which can lead to computational complexity.

2. **Backward State Space Planning (BSSP):**
    - BSSP functions related to backward state-space search.
    - It starts from the goal state and traces back through a series of actions to identify sub-goals or preconditions necessary for achieving the final goal.
    - This process, known as regression, involves moving from the target state to the sub-goals.
    - **Advantage:** Smaller branching factor compared to FSSP, reducing computational complexity.
    - **Disadvantage:** Not always sound, as inconsistency may arise.

Role of Planning in AI:

- Planning is essential for decision-making and action selection in AI systems.
- It involves determining the sequence of tasks to be performed by the system to accomplish a specific goal.
- Effective planning ensures efficient utilization of resources and optimal task execute

➢ Types of planning

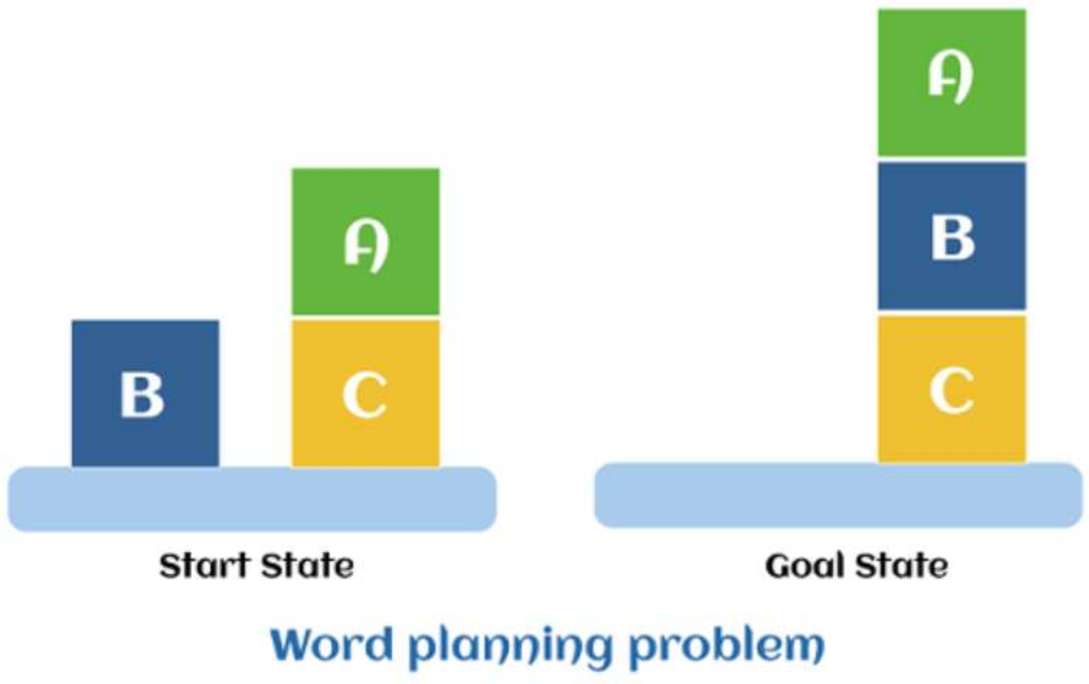
1. **Classical Planning:**
    - Classical planning involves creating a sequence of actions to achieve a predefined goal within a static and predictable environment.
    - Assumes a deterministic world where outcomes of actions are known in advance and do not change over time.
    - **Example:** Planning a route from point A to point B using a map where the road network is fixed.
2. **Hierarchical Planning:**
    - Hierarchical planning breaks down complex problems into smaller, more manageable sub-problems, organized in a hierarchical structure.
    - Involves creating a hierarchy of plans, with higher-level plans overseeing the execution of lower-level plans.
    - **Example:** Planning a trip involves higher-level plans such as booking flights and accommodations, which supervise lower-level plans like packing luggage and arranging transportation.
3. **Temporal Planning:**
    - Temporal planning considers time constraints and interdependencies between actions when creating plans for the future.
    - Ensures that plans are executable within specific time limits by accounting for the duration of tasks and their temporal relationships.
    - **Example:** Scheduling tasks in a project management tool, where dependencies and deadlines dictate the order and timing of activities.

➢ Components of the planning system:

1. **Rule Selection:** Choose the most appropriate rule based on available information and heuristics to guide the planning process effectively.
2. **Rule Application:** Apply the selected rule to update the problem condition, moving closer to the desired goal state.
3. **Solution Detection:** Identify when a solution has been found by reaching the target state or achieving specified criteria.

4. **Dead-End Detection:** Detect dead ends or infeasible paths to discard them and focus efforts on more promising directions.
5. **Optimality Detection:** Determine when a near-perfect solution is reached, balancing between computational resources and solution quality.

➢ Block-world planning problem:



- The block-world problem, also known as the **Sussmann anomaly**.
- It involves manipulating blocks labeled 'A', 'B', and 'C' on a flat surface to achieve a target configuration, with the constraint that *only one block can be moved at a time*.

Problem Description:

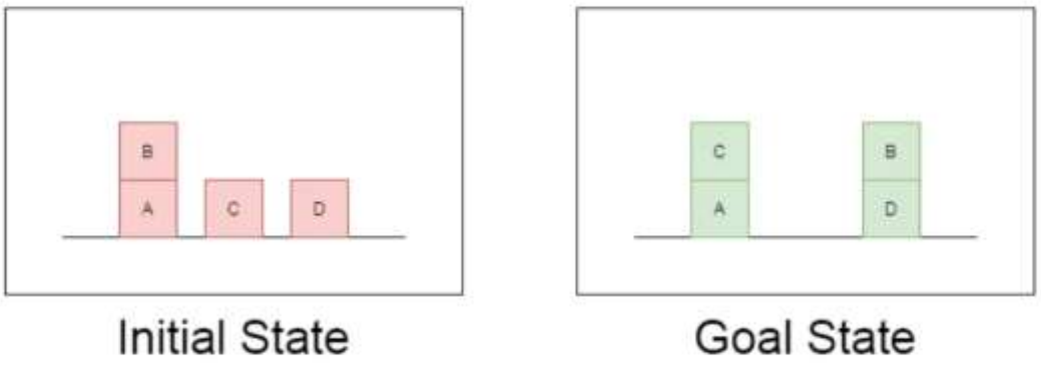
- Start Position: Blocks arranged in a specific configuration on the flat surface.
- Target Position: Desired arrangement of blocks to be achieved through a series of movements.

- Only one block can be moved at a time, and each movement must be carefully planned to reach the target configuration.

Challenge:

- Non-interleaved planners of the early 1970s struggled to solve the block-world problem efficiently due to its complexity.
- When faced with two sub-goals (G1 and G2), these planners often failed to produce a combined plan for both sub-goals, leading to an odd behavior.

➢ Goal stack planning:



Blocks World Problem:

The Blocks World Problem involves manipulating blocks on a table using a robot arm to achieve a desired configuration. The scenario typically includes:

- **Table:** A flat surface where blocks can be placed.
- **Blocks:** Objects that can be moved and stacked on the table.
- **Robot Arm:** A mechanism capable of picking up and putting down blocks one at a time.

- **Constraints:** Only one block can be moved at a time, and no block should be stacked on top of another during movement.

Goal Stack Planning:

Goal Stack Planning is an AI problem-solving method that works backward from the goal state to the initial state. Here's how it works:

1. **Initialization:** Start with the goal state and work backward toward the initial state.
2. Break down the goal into subgoals and recursively solve them.
3. Select actions to achieve subgoals based on preconditions and effects.
4. **Execution:** Execute selected actions to transform the world state closer to the goal state.
5. **Backtracking:** If a deadlock or inconsistency is encountered, backtrack and explore alternative paths.
6. **Termination:** Repeat steps until the initial state is reached, and the goal is achieved.

Components of Goal Stack Planning:

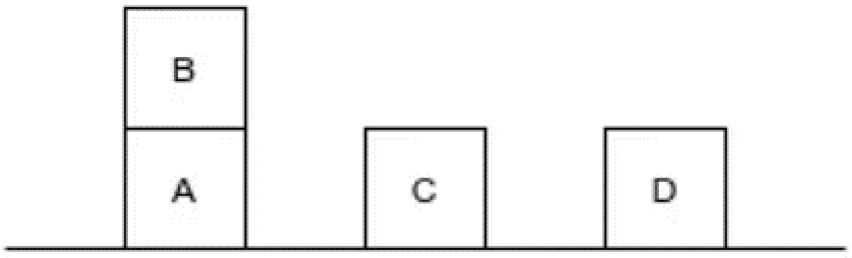
- **Initial State:** The starting configuration of blocks.
- **Goal State:** The desired configuration to be achieved.
- **World State:** Represents the current state of the environment during planning.
- **Predicates:** Statements conveying information about the configuration in the Blocks World, such as block positions and robot arm status.
- **Operations:** Actions performed by the robot arm, including picking up, putting down, stacking, and unstacking blocks.

Representing Configurations with Predicates:

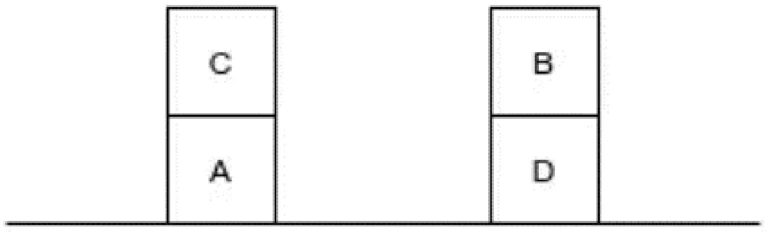
- **ON(A, B):** Block A is stacked on top of block B.
- **ONTABLE(A):** Block A is placed directly on the table.
- **CLEAR(A):** There is no other block stacked on top of block A.
- **HOLDING(A):** The robot arm is currently holding block A.
- **ARMEMPTY:** The robot arm is not holding any block.

Using these predicates, we can describe both the initial and goal states:

- **Initial State:**
  ON(B, A) ∧ ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ CLEAR(D) ∧ ARMEMPTY



- **Goal State:**
  ON(C, A) ∧ ON(B, D) ∧ ONTABLE(A) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ ARMEMPTY



Operations of the Robot Arm:

The robot arm can perform four operations, each with specific preconditions:

- **STACK(X, Y):** Stacking block X on top of block Y.
  - Preconditions: CLEAR(X), HOLDING(X), CLEAR(Y)
- **UNSTACK(X, Y):** Picking up block X from the top of block Y.
  - Preconditions: ON(X, Y), CLEAR(X), ARMEMPTY
- **PICKUP(X):** Picking up block X from the table.
  - Preconditions: ONTABLE(X), CLEAR(X), ARMEMPTY
- **PUTDOWN(X):** Placing block X on the table.
  - Preconditions: HOLDING(X)

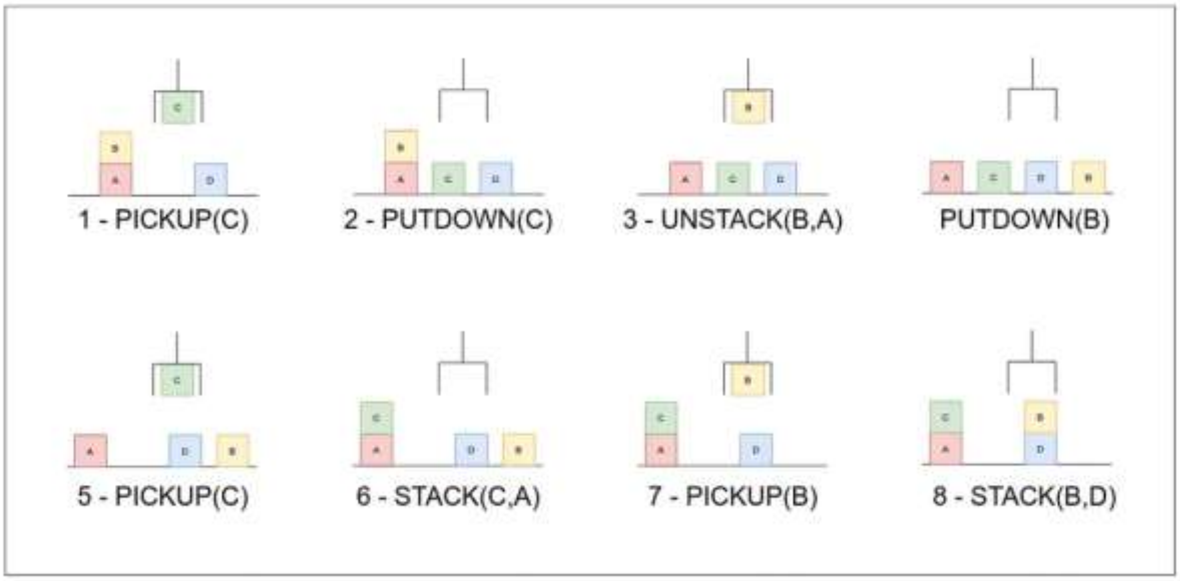
These operations enable the robot arm to manipulate blocks and rearrange them to achieve the desired goal state. Each operation requires certain conditions to be met before it can be executed successfully, ensuring the safety and effectiveness of block movements.

Solution:

- The effect of these operations is represented using two lists **ADD** and **DELETE**.
- DELETE List contains the predicates which will cease to be true once the operation is performed.

- ADD List on the other hand contains the predicates which will become true once the operation is performed.

OPERATORS	PRECONDITION	DELETE	ADD
STACK(X,Y)	CLEAR(Y)∧ HOLDING(X)	CLEAR(Y) HOLDING(X)	ARMEMPTY ON(X,Y)
UNSTACK(X,Y)	ARMEMPTY∧ ON(X,Y)∧ CLEAR(X)	ARMEMPTY∧ ON(X,Y)	HOLDING(X) ∧CLEAR(Y)
PICKUP(X)	CLEAR(X)∧ ONTABLE(X)∧ ARMEMPTY	ONTABLE(X)∧ ARMEMPTY	HOLDING(X)
PUTDOWN(X)	HOLDING(X)	HOLDING(X)	ONTABLE(X)∧ ARMEMPTY



➢ Non-linear planning:

- Non-Linear Planning in AI is a problem-solving approach used to address complex situations with unpredictable solutions.
- It involves finding a sequence of actions to transform the system's current state to the desired goal state, considering non-linear constraints like resource limitations and temporal ordering.

- **Algorithm:**

1. **Select Goal:**
    - Choose a goal, denoted as 'g', from the set of goals.
2. **Operator Selection:**
    - If the current state does not match goal 'g', select an operator 'o' whose add-list matches goal 'g'.
3. **Push Operator:**
    - Push operator 'o' onto the OpenStack and add the preconditions of 'o' to the set of goals.
4. **Continue Process:**
    - Continue this process while all preconditions of the operator on top of the OpenStack are satisfied in the current state.
5. **Apply Operator:**
    - Pop the operator 'o' from the top of the OpenStack, apply it to the current state, and add it to the plan.

Advantages:

- **Flexibility:** Handles complex problem spaces with interdependent sub-goals and non-linear dependencies.
- **Optimality:** Utilizes heuristic search techniques to find optimal solutions.
- **Resource Utilization:** Considers resource limitations and temporal ordering constraints.
- **Reusability:** Can reuse previously generated plans for similar problems, improving efficiency.
- **Integration:** Can be integrated with other AI techniques like machine learning for robust problem-solving.

Disadvantages:

- **Complexity:** Dealing with complex and interconnected sub-goals can increase computational overhead.
- **Scalability:** May face challenges with large problem spaces, where the search space becomes too large to explore efficiently.
- **Uncertainty:** Assumes a deterministic world, which may not always hold true in real-world scenarios.

- **Domain Expertise:** Relies on domain expertise to generate effective plans, which may require significant knowledge.