PLANNING

Module 4

Planning

- Planning is a fundamental property of intelligent behavior.
- It is the first and foremost activity to achieve desired results.
- The planning in Artificial Intelligence is about the decision making tasks performed by the robots or computer programs to achieve a specific goal.
- The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task

- Planning is the process of decomposing a problem into appropriate subparts and of recording and handling interactions among the subparts as they are detected during the problem solving process.
- The word planning also refers to the process of computing several steps of a problem solving procedure before executing them

Decomposition of Problems

- When trying to solve a complicated problem, it may be necessary to work on small pieces of the problem separately and then combine the partial solutions at the end into a complete problem solution.
- The decomposition may divide a single difficult problem into several easier sub-problems.

Nearly Decomposable Problems

- It may not be always possible to divide a problem into completely separate sub-problems.
- However, problems may be nearly decomposable by which we mean that they can be divided into sub-problems that have only a small amount of interaction

Components of a planning system

- The following are the components of a planning system:
  1. Choosing rules to apply
  2. Applying rules
  3. Detecting a solution
  4. Detecting dead ends
  5. Repairing an almost correct solution

1. Choosing rules to apply

The commonly used technique can be described as:

1. Isolate a set of differences between the desired goal state and the current state.
2. Identify those rules that reduce the differences.
3. If several rules are found, use heuristic methods to choose one.

2. Applying Rules

- In simple systems, application of rules is easy. Each rule simply specified the problem state that would result from the application of the rule.
- In complex systems rules specify only a small part of the problem state. In such cases there are several methods for the application of rules.

2. Applying Rules

1. Describe each of the changes to the state description the application of the rule makes. Add a new rule that everything else is remaining unchanged. To do this, we may require the explicit statements of a large number of premises or axioms.
2. Describe each operator by a list of new predicates (called ADD) that the operator causes to become true and a list of old predicates (called DELETE) that the operator causes to become false. There is also a third list (called PRECONDITION) which specifies the predicates that must be true for the operator to be applied. Any predicate not included in either ADD or DELETE is is assumed to be unaffected by the rule.

3. Detecting a Solution

- A planning system can be considered to have succeeded in finding a solution to a problem when it has found a sequence of operators that transform the initial problem state into the goal state.
- In simple problems the system can easily determine whether it has succeeded in finding a solution by checking whether the problem state and goal state match.

3. Detecting a Solution

- But in complex problems, in a description of a problem state, the entire state may not be explicitly described. So in such cases there should be a method to determine whether two problem states match. This method will depend on how the various states are represented in the planning system.
- For example, suppose that a planning system is represented using predicate logic. Let there be a predicate P(x) as part of the goal. T o see whether P(x) has been satisfied at any stage, it is enough to prove that we can derive P(x) from the assertions describing that particular state. If we can construct such a proof, the problem-solving process can be ended

4. Detecting dead-ends

- The planning system must be able to detect dead-end, that is, it must be able to determine whether a path it is exploring can never lead to a solution.
- The mechanisms for detecting a solution are used to detect dead-ends also

5. Repairing an almost correct solution

- Consider the problem of solving a nearly decomposable problem.
- We assume that the problem is completely decomposable and solve each sub-problem separately and then combine the solutions of the sub-problems to obtain a solution of the original problem.
- Since the problem is only nearly decomposable, the combined solution may not be the correct solution; it will only be an almost correct solution.

5. Repairing an almost correct solution

- The following are some of the ways in which an almost correct solution may be repaired.

1. Throw out the almost correct solution and look for another.
2. Consider the situation arising from the execution of the sequence of operations specified in the proposed almost correct solution. The difference between this and the goal will much less than the difference between the initial state and the goal. The problem solving system may be applied again to eliminate the difference.
3. Find exactly what went wrong and try a direct patch

Goal Stack Planning

- In goal stack planning, we make use of stack called a goal stack usually denoted by GS.
- The goal stack GS contains both subgoals and actions that have been proposed to satisfy those subgoals.
- It relies on a database, denoted by DB, that describes the current situation, and a set of actions described by PRECONDITION, ADD and DELETE lists

1. Push the compound predicate describing the goal state in to the Stack.
2. Push the individual predicates of the goal state in to the Stack.
3. Loop till the Stack is empty
    (a) Pop an element E from the Stack.
    (b) If E is a predicate
        i. If E is True
            A. Do nothing.
        ii. Else
            A. Push the relevant action into the Stack.
            B. Push the individual predicates of the precondition of the action into the Stack.
    (c) Else if E is an action "a"
        i. Apply the action "a" to the current state.
        ii. Add the action "a" to the plan.

The STRIPS planner

- The reasoning strategy used by STRIPS is goal stack planning. STRIPS (Stanford Research Institute Problem Solver) is an automated planner developed by Richard Fikes and Nils Nilsson in 1971.

Illustration: Blocks world

1. The objects of the blocks world

- A flat surface on which blocks can be placed.
- A number of square blocks, all of the same size.
- A robot arm that can manipulate the blocks. Blocks can be stacked one upon another.

Goal Stack Planning: Blocks World Problem

2. Predicates and relations to describe the state of the blocks world

Predicate	**Meaning**
ON(x, y)	Block x is on block y
ONTABLE(x)	Block x is on table.
CLEAR(x)	There is nothing on top of block x
HOLDING(x)	The arm is holding block x
ARMEMPTY	The arm is holding nothing.

Goal Stack Planning: Blocks World Problem

3. Actions

The following are the available actions in the blocks world.

Action	Meaning
UNSTACK(x, y)	Pick up block x from its current position on block y and hold it in the arm
STACK(x, y)	Place block x held in the arm on block y.
PICKUP(x)	Pick up block x from table and hold it.
PUTDOWN(x)	Put block x held in the arm down on the table.

Goal Stack Planning: Blocks World Problem

Action	PRECONDITION list	ADD list	DELETE list
UNSTACK(x, y)	ARMEMPTY, CLEAR(x), ON(x, y)	HOLDING(x), CLEAR(y)	ARMEMPTY
STACK(x, y)	HOLDING(x), CLEAR(y)	ARMEMPTY, ON(x, y), CLEAR(x)	
PICKUP(x)	ARMEMPTY, CLEAR(x).	HOLDING(x)	ARMEMPTY
PUTDOWN(x)	HOLDING(x).	ARMEMPTY, ONTABLE(x), CLEAR(x)	

- Given the initial and the final states of a blocks world with four blocks, use goal stack planning algorithm to obtain a plan for achieving the goal state

Figure 8.2: A simple blocks world problem: Initial state on the left and the goal state on the right

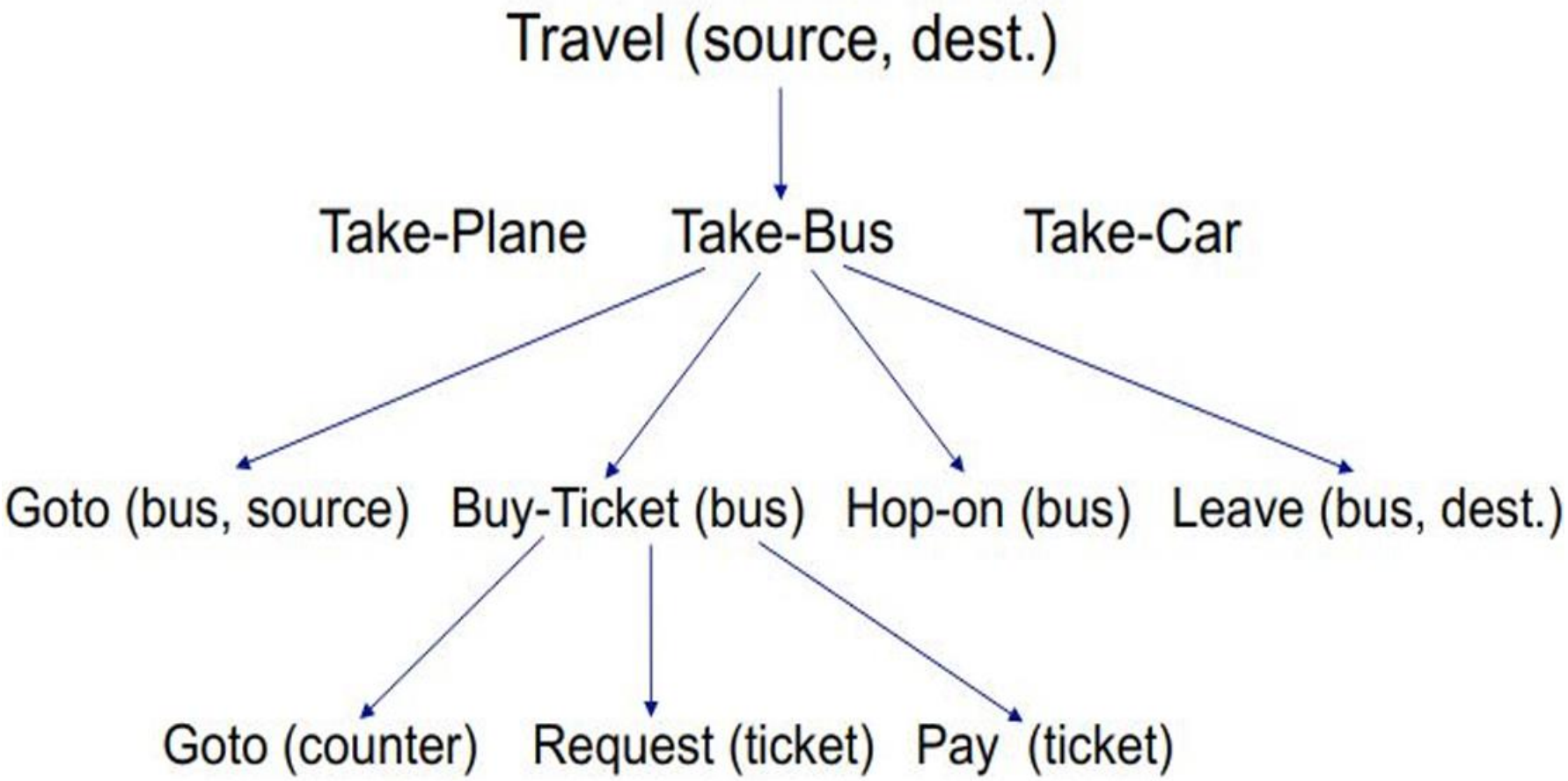
Hierarchical planning

- In order to solve real world problems, the problem solver may have to generate long plans.
- To do this efficiently, the problem solver may ignore some of the details of the problem until a solution to the main issues have been found.
- This can be done by creating a hierarchy of the tasks to be performed and then attempting to find a solution starting from the tasks at the highest level of the hierarchy.
- Creating a plan in this way is known as hierarchical planning

Hierarchical planning-Example

- For example, let it be required to travel at short notice to Mumbai with minimum expenses for an interview.
- The task of the highest priority is to check the availability of an airline at affordable cost.
- After finding a plan to solve this problem, the person may find plans for solving the next level problems like preparing baggage and hiring a taxi.
- Attempts to solve these problems lead to problems at the third level: detailed plans for preparing the baggage and hiring a taxi.
- These will in turn lead to problems at the next lower level.

Hierarchical Plan to travel from a certain Source to Destination by Bus



The ABSTRIPS planner system

- ABSTRIPS, a planning system developed in 1974 and built on top of the STRIPS planning system, is a planning system has a method for the implementation of hierarchical planning.
- In ABSTRIPS, the hierarchies are constructed by assigning criticalities which are numbers indicating relative difficulty to the preconditions of the operator.

The ABSTRIPS planner system

- First a plan is found that satisfies only the preconditions of the operators of the highest criticality values.
- The plan is then refined by considering the preconditions at the next level of criticality and inserting steps into the plan to achieve these preconditions.
- The process is repeated to the lowest level of criticality.
- For this system to work, there must be a scheme to assign the appropriate criticality values to the various terms in the precondition.
- Because this process explores entire plans at one level of detail before looking any of the lower details, it has been called length-first search.