Unit-7: Planning

Computer Engineering Department

Outline

- Introduction
- The Blocks World
- Components of a Planning System
- Goal Stack Planning
- Nonlinear Planning Using Constraint Posting
- Hierarchical Planning
- Reactive Systems

Introduction

- 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 task of finding a procedural course of action for a declaratively described system to reach its goals while optimizing overall performance measures.
- Automated planners find the transformations to apply in each given state out of the possible transformations for that state.
- In contrast to the classification problem, planners provide guarantees on the solution quality.
- Automation is an emerging trend that requires efficient automated planning.
- Many applications of planning in industry are, e.g.. robots and autonomous systems, cognitive assistants, cyber security, service composition, etc.

Introduction

- Requirements of AI Planning Techniques
  - When explainability is desired
  - When you want to be able to explain why a particular course of action was chosen
  - Assignment of responsibility is essential for automation of processes (e.g., autonomous driving, medical expert systems)

- In many real life applications, there is a structure of the problem that cannot be learned with Deep Learning (DL).
- Solving optimization problems with learning is hard, but integrating planning techniques with heuristic guidance learned by DL will result in the most famous success stories of AI to day.
  - GO player AlphaGO uses planning (monte-carlo tree search) with deep learning (heuristic guidance) to select the next move
  - Cognitive assistant (Samsung) uses knowledge graph, planning, and deep learning to answer complicated queries

Components of a Planning System

- Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.
- The planning consists of following important steps:

1. Choose the best rule for applying the next rule based on the best available heuristics.
   - The most widely used technique for selecting appropriate rules to apply is first to isolate a set of differences between desired goal state and then to identify those rules that are relevant to reduce those differences.
   - If there are several rules, a variety of other heuristic information can be exploited to choose among them.
2. Apply the chosen rule for computing the new problem state.
   - In simple systems, applying rules is easy. Each rule simply specifies the problem state that would result from its application.
   - In complex systems, we must be able to deal with rules that specify only a small part of the complete problem state.
   - One way is to describe, for each action, each of the changes it makes to the state description.

Components of a Planning System

3. Detect when a solution has been found.
   - A planning system has 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.
   - How will it know when this has done?
   - In simple problem-solving systems, this question is easily answered by a straightforward match of the state descriptions.
   - One of the representative systems for planning systems is predicate logic. Suppose that as a part of our goal, we have the predicate P(x).
   - To see whether P(x) satisfied in some state, we ask whether we can prove P(x) given the assertions that describe that state and the axioms that define the world model.

Components of a Planning System

4. Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.
   - As a planning system is searching for a sequence of operators to solve a particular problem, it must be able to detect when it is exploring a path that can never lead to a solution.
   - The same reasoning mechanisms that can use to detect a solution can often use for detecting a dead end.
   - If the search process is reasoning forward from the initial state. It can prune any path that leads to a state from which the goal state cannot reach.
   - If search process reasoning backward from the goal state, it can also terminate a path either because it is sure that the initial state cannot reach or because little progress made.

5. Detect when an almost correct solution has been found.
   - The kinds of techniques discussed are often useful in solving nearly decomposable problems.
   - One good way of solving such problems is to assume that they are completely decomposable, proceed to solve the sub-problems separately. And then check that when the sub-solutions combined.
   - They do in fact give a solution to the original problem.

Blocks World Problem

- Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.
- In order to compare different methods of planning, we should look at all of them in a single domain.
- The Block World Problem is described as,
  - There is a flat surface on which blocks can be placed.
  - There are a number of square blocks, all the same size.
  - The blocks are labeled with alphabets ‘A’, ‘B’, ‘C’, etc.
  - They can be stacked one upon the other.
  - There is robot arm that can manipulate the blocks.
  - The start state and goal state are given.

Blocks World Problem

- Actions of the robot arm (the robot arm can hold only one block at a time)
  - UNSTACK(A, B): Pick up block A from its current position on block B.
  - STACK(A, B): Place block A on block B.
  - PICKUP(A): Pick up block A from the table and hold it.
  - PUTDOWN(A): Put block A down on the table.

- Predicates : In order to specify both the conditions under which an operation may be performed and the results of performing it, we need the following predicates:

1. ON(A, B): Block A is on Block B.
2. ONTABLES(A): Block A is on the table.
3. CLEAR(A): There is nothing on the top of Block A.
4. HOLDING(A): The arm is holding Block A.
5. ARMEMPTY: The arm is holding nothing.

STRIPS-Based Approach to Robot Control

- It uses the first-order logic and theorem proving to plan strategies from start to goal.
- STRIPS language: “Classical” approach that most planners use, lends itself to efficient planning algorithms.
- Environment: office environment with specially colored and shaped objects.
- STRIPS planner: developed for this system to determine the actions of the robot should take to achieve goals.
- STRIPS (STanford Research Institute Problem Solver) is a restrictive way to express states, actions and goals, but leads to more efficiency.

Robot Problem-Solving Systems (STRIPS)

- ADD List : List of new predicates that the operator causes to become true.
- DELETE List : List of old predicates that the operator causes to become false.
- PRECONDITIONS list contains those predicates that must be true for the operator to be applied.
- STRIPS style operators for BLOCK World problem are :

  - STACK(x, y)
  - P: CLEAR(y) Λ HOLDING(x)
  - D: CLEAR(y) Λ HOLDING(x)
  - A: ARMEMPTY Λ ON(x, y)
  - UNSTACK(x, y)
  - PICKUP(x)
  - P: CLEAR(x) Λ ONTABLE(x) Λ ARMEMPTY
  - D: ONTABLE(x) Λ ARMEMPTY
  - A: HOLDING(x)
  - PUTDOWN(x)

Goal Stack Planning

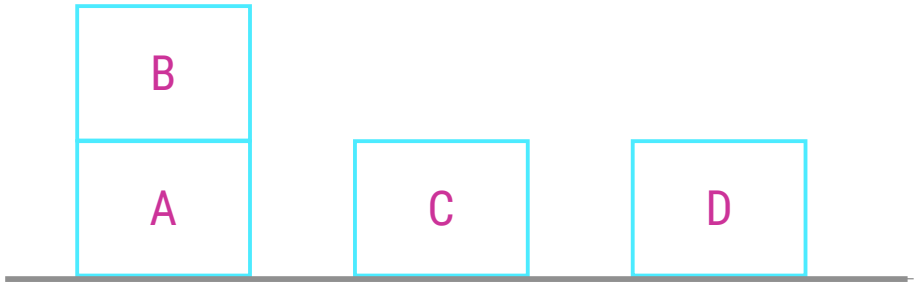
- Goal Stack Planning is the one of the simplest planning algorithms that is designed to handle problems which include compound goals.
- It utilizes STRIPS as a formal language for specifying and manipulating the world with which it is working.
- This approach uses a Stack for plan generation. The stack can contain Sub-goal and actions described using predicates. The Sub-goals can be solved one by one in any order.
- It starts by pushing the unsatisfied goals into the stack.
- Then it pushes the individual sub-goals into the stack and its pops an element out of the stack.
- When popping an element out of the stack the element could be,
  - either a predicate describing a situation about our world or
  - it could be an action that can be applied to our world under consideration.

Goal Stack Planning

- So, a decision has to be made based on the kind of element we are popping out from the stack.
- If it is a Predicate, then compares it with the description of the current world, if it is satisfied or is already present in our current situation then there is nothing to do because already its true.
- On the contrary if the Predicate is not true then we have to select and push relevant action satisfying the predicate to the Stack.
- So after pushing the relevant action into the stack its precondition should also has to be pushed into the stack.
- In order to apply an operation its precondition has to be satisfied, i.e., the present situation of the world should be suitable enough to apply an operation.
- For that, the preconditions are pushed into the stack once after an action is pushed.

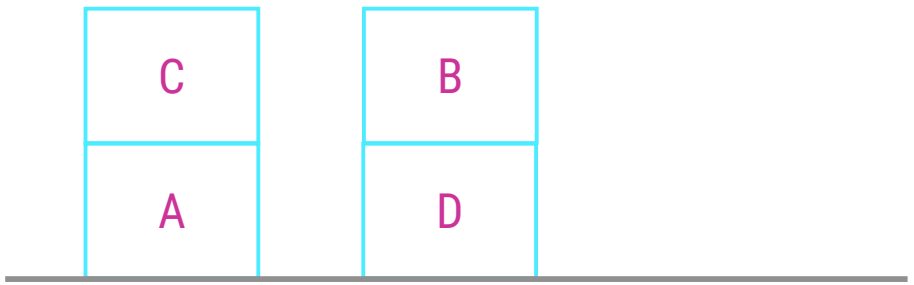
Goal Stack Planning – Example

- Lets start here with the BLOCK WORLD example, the initial state is our current description of our world. The Goal state is what we have to achieve.



Initial State

ON(B, A)ΛONTABLE(C) ΛONTABLE(A) ΛONTABLE(D) ΛCLEAR(D) ΛCLEAR(C) ΛCLEAR(B) Λ ARMEMPTY



Goal State

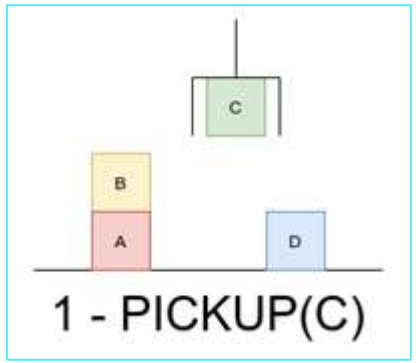
ON(C,A)ΛON(B,D) ONTABLE(A) ΛONTABLE(D) ΛCLEAR(C) ΛCLEAR(B) Λ ARMEMPTY

Goal Stack Planning – Example

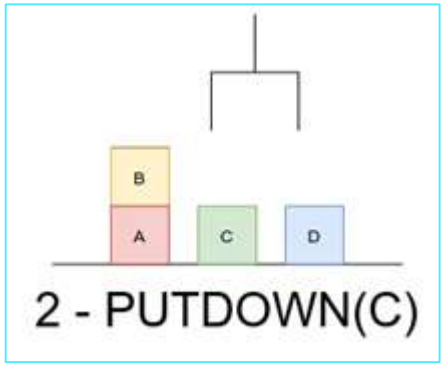
- The following list of actions can be applied to the various situations in the problem.

OPERATORS	PRECONDITION	DELETE	ADD
STACK(A, B): Place block A on block B.	CLEAR(B) ∧ HOLDING (A)	CLEAR(B) ∧ HOLDING (A)	ARMEMPTY ∧ ON(A,B)
UNSTACK(A, B): Pick up block A from its current position on block B.	ON(A,B) ∧ CLEAR(A) ∧ ARMEMPTY	ON(A,B) ∧ ARMEMPTY	HOLDING (A) ∧ CLEAR(B)
PICKUP(A): Pick up block A from the table and hold it.	CLEAR(A) ∧ ONTABLE(A) ∧ ARMEMPTY	ONTABLE(A) ∧ ARMEMPTY	HOLDING (A)
PUTDOWN(A): Put block A down on the table.	HOLDING (A)	HOLDING (A)	ONTABLE(A) ∧ ARMEMPTY

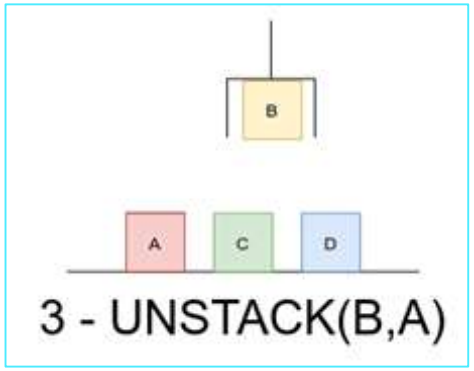
Goal Stack Planning – Example



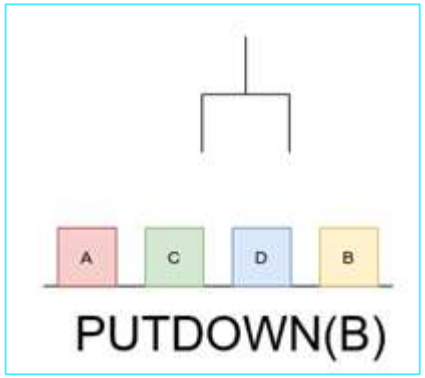
1 - PICKUP(C)



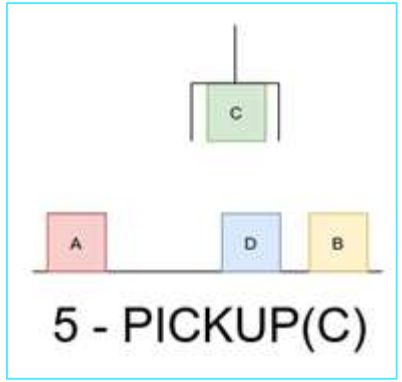
2 - PUTDOWN(C)



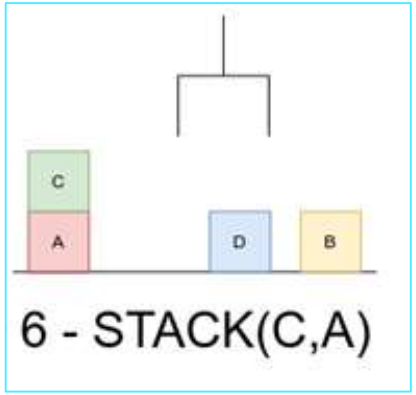
3 - UNSTACK(B,A)



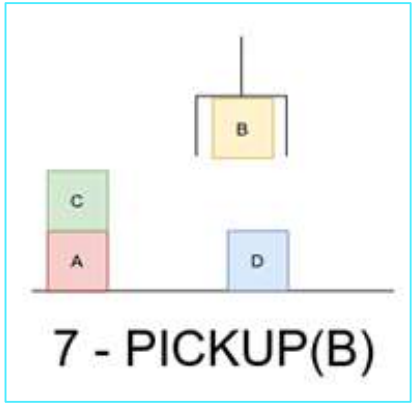
PUTDOWN(B)



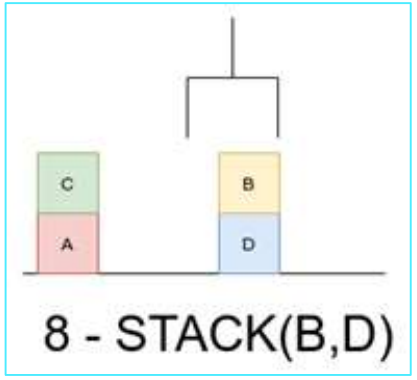
5 - PICKUP(C)



6 - STACK(C,A)



7 - PICKUP(B)



8 - STACK(B,D)

Nonlinear Planning using Constraint Posting

- This planning is used to set a goal stack and is included in the search space of all possible sub-goal orderings. It handles the goal interactions by interleaving method.
- Many problems require an intertwined plan in which multiple sub-problems worked on simultaneously.
- Such a plan is called nonlinear plan because it is not composed of a linear sequence of complete sub-plans.
- Non-linear planning may be an optimal solution with respect to plan length (depending on search strategy used).
- It takes larger search space, since all possible goal orderings are taken into consideration.

Constraint Posting

- The idea of constraint posting is to build up a plan by incrementally hypothesizing operators, partial orderings between operators, and binding of variables within operators.
- Constraint posting often comes with Non-Linear Planning. The idea of constraint posting is to build up a plan incrementally.
- At any given time in the problem-solving process, we may have a set of useful operators but perhaps no clear idea of how those operators should order with respect to each other.
- A solution is a partially ordered, partially instantiated set of operators to generate an actual plan. And we can convert the partial order into any number of total orders.

Algorithm: Nonlinear Planning

1. Choose a goal 'g' from the goalset
2. If 'g' does not match the state, then
   - ➥ Choose an operator 'o' whose add-list matches goal g
   - ➥ Push 'o' on the opstack
   - ➥ Add the preconditions of 'o' to the goalset
3. While all preconditions of operator on top of opstack are met in state
   - ➥ Pop operator o from top of opstack
   - ➥ state = apply(o, state)
   - ➥ plan = [plan; o]

Hierarchical Planning

- In order to solve hard problems, a problem solver may have to generate long plans.
- It is important to be able to eliminate some of the details of the problem until a solution that addresses the main issues is found.
- Then an attempt can be made to fill in the appropriate details.
- Early attempts to do this involved the use of macro operators, in which larger operators were built from smaller ones.
- Lower level activities would detail more precise steps for accomplishing the higher level tasks.
- Instead of having to try out a large number of possible plan ordering, plan hierarchies limit the ways in which an agent can select and order its primitive operators.

Hierarchical Planning

- Hierarchical Planning describes Hierarchy of actions in terms of major action or minor action.
- Planning for ”Going to Goa this Christmas”
  - Major Steps :
    - Hotel Booking
    - Ticket Booking
    - Reaching Goa
    - Staying and enjoying there
    - Coming Back
  - Minor Steps :
    - Take a taxi to reach station / airport
    - Have candle light dinner on beach
    - Take photos

Hierarchical Planning

- Hierarchical Planning Properties
  - Postpone attempts to solve mere details, until major steps are in place.
  - Identify a hierarchy of conditions
  - Construct a plan in levels, postponing details to the next level
  - Patch higher levels as details become visible
- Hierarchy of conditions reflect the intrinsic difficulty of achieving various conditions. Intrinsic difficulty is indicated by criticality value.
- A operation having minimum criticality can be trivially achievable, i.e., the operations having very less or no precondition. For example : Opening makemytrip.com
- Similarly operation having many preconditions to satisfy will have higher criticality.
- The assignment of appropriate criticality value is crucial to the success of this hierarchical planning method.
- Those preconditions that no operator can satisfy are clearly the most critical.

ABSTRIPS : Modified version of STRIPS that incorporates hierarchical planning

- A better approach is developed in ABSTRIPS systems which is actually planned in a hierarchy of abstraction spaces, in each of which preconditions at a lower level of abstraction are ignored.
- ABSTRIPS (Abstraction-Based STRIPS) approach is as follows:
  - First solve the problem completely, considering only preconditions whose criticality value is the highest possible.
  - These values reflect the expected difficulty of satisfying the precondition.
  - To do this, do exactly what STRIPS did, but simply ignore the preconditions of lower than peak criticality.
  - Once this done, use the constructed plan as the outline of a complete plan and consider preconditions at the next-lowest criticality level.
  - Augment the plan with operators that satisfy those preconditions.
- Patching in ABSTRIPS : Each level starts with the goal stack that includes the plan obtained in the higher levels. The last item in the goal stack being the main goal.
- This approach explores the entire plans at one level of detail before it looks at the lower-level details of any one of them, so, it is called length-first approach.

Reactive Systems

- The idea of reactive systems is to avoid planning altogether, and instead, use the observable situation as a clue to which one can simply react.
- A reactive system must have an access to a knowledge base of some sort that describes what actions should be taken under what circumstances.
- A reactive system is very different from the other kinds of planning systems we have discussed. Because it chooses actions one at a time.
- It does not anticipate and select an entire action sequence before it does the first thing.
- The example is a Thermostat. The job of the thermostat is to keep the temperature constant inside a room.
- Reactive systems are capable of complex behaviors.
- The main advantage reactive systems have over traditional planners is that they operate robustly in domains that are difficult to model completely and accurately.

Reactive Systems

- Reactive systems dispense with modeling altogether and base their actions directly on their perception of the world.
- Another advantage of reactive systems is that they are extremely responsive since they avoid the combinatorial explosion involved in deliberative planning.
- This makes them attractive for real-time tasks such as driving and walking.

Other Planning Techniques

- Triangle tables
  - Provides a way of recording the goals that each operator expected to satisfy as well as the goals that must be true for it to execute correctly.
- Meta-planning
  - A technique for reasoning not just about the problem solved but also about the planning process itself.
- Macro-operators
  - Allow a planner to build new operators that represent commonly used sequences of operators.
- Case-based planning:
  - Re-uses old plans to make new ones.

Thank You!