32002: AI (Planning) **Spring 2014**

Lecture 36: April 15, 2014

Instructor: K.R. Chowdhary *: Professor of CS*

36.1 Planning

Interest in automated planning is at least as old as artificial intelligence. A solution to a problem can be described in terms of a sequence of steps that transforms some initial description of the problem state, for example, the initial configuration of a puzzle, into a description satisfying a specified goal criterion. The steps (transformations) are called *operators* and a problem is defined in terms of a set of operators and a language for describing problem states. In automated planning, the problem states correspond to instantaneous descriptions of the world and operators correspond to *actions* that an agent can perform to change the state of the world.

The planning is about how an agent achieves its goals. To achieve even the simplest goals an agent must reason about the future. Since the goal is not achievable in single step, the number of steps to be carried out needs to be broken up into subtasks, and steps for each needs t be the goal, what an agent will do in next step also depends on its past.

To complete each subtask, there are actions, to be carried out, each action has a subsequent state as well as a preceding state. To be simple at start of this subject, we make certain assumptions:

1. the actions are deterministic, i.e., the agent can determine the consequent of the actions,
2. the world is fully observable, i.e, the agent can observe the correct state of the world, and
3. the closed world assumption, .i.e, the fact snot described in the world are false.

36.2 The Basic Planning Problem

A basic planning problem usually comprises an initial world description, a description of the goal world, and a set of actions (sometimes also called *operators*) that map a world description to another. A solution is a sequence of actions leading from the initial world description to the goal world description, referred to as a *plan*.

A deterministic action is a *partial function* from states to states; the partial because for every state "(state, action)" pair does not necessarily result to a state. Hence, it is not a total function. For example, a robot can move block x onto y, if x has top-clear, y has top-clear, and obviously, $x \neq y$. A *precondition* of an action decides about when the action can be carried out, and resulting state due to an action is the effect of the actions.

The general structure of a planning problem is straight forward: (the relevant part of) the world is in a certain state, but managers or directors would like it to be in another state. The (abstract) problem of how

one should get from the current state of the world through a sequence of actions to the desired goal state is a planning problem.

AI planning techniques are techniques to search for a plan: *forward planning* is a planning technique building a plan starting from the initial state; *backward planning* starts from the goal states, and *least-commitment planning* constructs plans by adding actions in a non-sequential order.

Ideally, to solve such planning problems, we would like to have a general planning-problem solver. However, such an algorithm, solving all planning problems, can be proven to be non-existing (That is, the general planning problem is undecidable). We therefore try to concentrate on a simplification of the general planning problem called *the classical planning problem*. Although not all realistic problems can be modeled as a classical planning problem, they can help to solve more complex problems.

36.2.1 The Classical Planning Problem

The classical planning problem can be defined as follows. Given the,

1. a description of the known part of the *initial state* of the world (in a formal language, usually propositional logic) denoted by **I**,
2. a description of the *goal* (i.e., a set of goal states), denoted by **G**, and
3. a description of the possible **atomic actions** (**R** (rule))that can be performed, modeled as state transformation functions,

determine a plan, i.e., a sequence of actions that transforms each of the states fitting the initial configuration of the world into one of the goal states.

Example 36.1 *Transport by taxi.*

Suppose that initially (i.e., in all states of the world that match the description **I**), there is a taxi at a location A, represented by a binary state variable $taxi(A)$, and a passenger at a location B, represented by $passgr(B)$. In each of the states described by **G** the passenger should be at a location C, denoted by $passgr(C)$. Furthermore, suppose that there are three actions that can transform (some part of) the state of the world.

Following are the steps for actions:

1. The taxi can move from one location to another: $move(x, y)$ with $x, y \in \{A, B, C\}$. This action requires that a priori $taxi(x)$ holds, and ensures that in the resulting state $\neg taxi(x)$ and $taxi(y)$ hold.
2. The passenger can get into the taxi: $load(passgr)$. This action requires a priori $taxi(x)$ and $passgr(y)$ and $x = y$, and in the resulting state both $\neg passgr(y)$ and $passgr(taxi)$ (i.e., passenger in taxi) should hold.
3. The passenger can get out of the taxi: $unload()$. This action requires that taxi is at location x ($taxi(x)$) and passenger in taxi ($passgr(taxi)$), and results in $\neg passgr(taxi)$ and $passgr(x)$.

With **I** as initial set of states, and **G** as set of goal states, the sequence of state transitions can indicated as follows:

I	;Initial state
move(A, B)	; taxi moves from location A to B
load(passgr)	; passenger gets into taxi
move(B, C)	; taxi moves from location B to C
unload(passgr)	; passenger unloads from taxi
G	; goal: taxi at C

□

36.2.2 Agent types

Agents can be classified according to the techniques they employ in their decision making:

1. *reactive agents*, which base their next decision solely on their current sensory input,
2. *planning agents*, which do this by taking into consideration, the anticipated future situations, possibly as a result of their own actions, to decide on the best course of action.

Whether an agent should plan or it should be reactive, depends on the particular situation it finds itself in. Consider the case where an agent has to plan a route from one place to another. A *reactive* agent might use a compass to plot its course, whereas a *planning agent* would consult a map. Clearly, the planning agent will come up with the shortest route in most cases, as it will not be confronted with uncrossable rivers and one-way hills. On the other hand, there are also situations where a reactive agent can be at least as effective, for instance if there are no maps to consult such as in a domain of planetary exploration, like Mars or Moon. Nevertheless, the ability to plan ahead is invaluable in many domains.

Considering the states set as $s_1, s_2, \ldots$, and set of actions as $a_1, a_2, \ldots,$, they are be represented sing a tree shown in figure 36.1 or explicitly by a table 36.1.

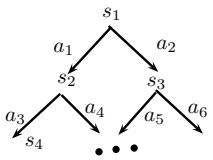

Figure 36.1: World States and Robot Actions.

Table 36.1: Table for mapping: State × Action → State.

State	Action	Resulting State
s_1	a_1	s_2
s_1	a_2	s_3
s_2	a_2	s_4
s_2	a_3	s_5
...	...	...

Example 36.2 *A delivery Robot system (Robo) to deliver* Mail *and* Coffee *(Figure 36.2).*

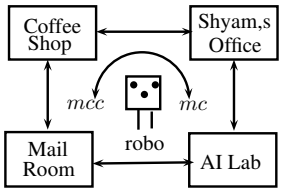

Figure 36.2: Delivery Robot with service locations.

A robot, called *Robo* can buy coffee at coffee-shop, pickup mail in the mail-room, and can move from one location to other by actions *mc* (move clockwise) and *mcc* (move counter clockwise). The domain of the world is represented by the terminology described as follows.

Various locations are represented by following symbols:

cs	Coffee shop
off	Shyam's office
mr	mail room
lab	AI lab

Various states of the world are:

rhc	Robo is holding coffee
swc	Shyam wants coffee
rhm	Robo is holding mail

Various actions performed by the Robo are:

mc	Robo moves clockwise
mcc	Robo moves counter clockwise
puc	Pickup coffee coffee
dc	Deliver coffee
pum	Pickup Mail
dm	Deliver Mail

A state may comprise many parameters or preconditions for the action to take place at that state. For example,

$$< lab, \overline{rhc}, swc, \overline{mw}, rhm >$$

indicate that Robo is in AI lab, Robo has no coffee in hand, Shyam wants coffee, mail is not waiting, and Robot holds Mail. Another state,

$$< lab, rhc, swc, mw, \overline{rhm} >$$

indicates that Robo is in lab, Robo is holding coffee, Shyam is waiting for coffee, mail is waiting, and Robo has no mail. The table 36.2 table shows the transitions for certain states.

Table 36.2: Some mapping: State × Action → State, for fig. 36.2.

State	Action	Resulting state
$< lab, \overline{rhc}, swc, \overline{mw}, rhm >$	mc	$< mr, \overline{rhc}, swc, \overline{mw}, rhm >$
$< lab, \overline{rhc}, swc, \overline{mw}, rhm >$	mcc	$< off, \overline{rhc}, swc, \overline{mw}, rhm >$
$< off, \overline{rhc}, swc, \overline{mw}, rhm >$	dm	$< off, \overline{rhc}, swc, \overline{mw}, \overline{rhm} >$
...	...	...

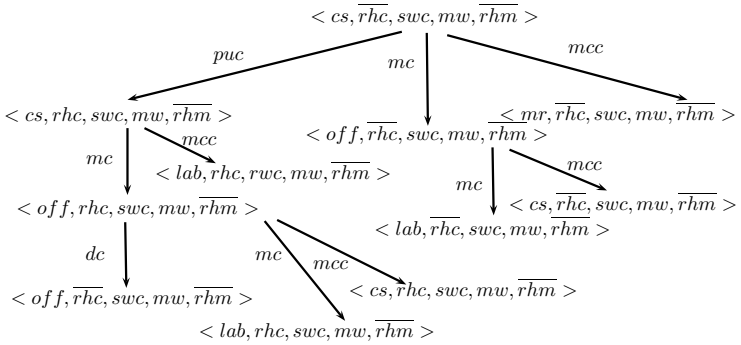

Figure 36.3: State-Space for Forward Planning.

36.2.3 Forward Planning

It is one of the simplest planning to treat the planning problem as a path planning problem in the state-space graph. The nodes here are states, transitions are actions, and results of actions are also states. A forward planner searches the state-space graph from start for goal state. The figure 36.3 shows the state-space graph for forward planning with start state as $< cs, \overline{rhc}, rhc, mw, \overline{rhm} >$, and three transitions from start state, corresponding to actions: pickup coffee (puc), Robo moves clock-wise (mc), and Robo moves counter-clockwise (mcc).

The branching factor in figure 36.3 is 3, and the search can be done in DFS or BFS. Theoretically, since, the Robo can be at any of the four locations, and other four parameters in the state can be true / false, there are $4 \times 2 \times 2 \times 2 \times 2 = 64$ total possible states in the world. Obviously, all of these states are not possible to reach.

The representation above is simple and clear, but it is not good due to following reasons:

- there are too many states to acquire, reason, and represent,
- small change in the requirements will need a major change in the model. For example, if need to have information about battery to be added as one of the parameter, the entire structure gets modified.

The improvement in complexity is possible, and can be based on the following criteria: we note that in the actions there is structure and that can be used to make actions compact. We also note that a precondition of an action should be true before the action takes place. For example, the action of Robo to pickup the coffee (puc) requires the precondition of "Robo's location is coffee shop and Robo does not hold the coffee", which is expressed as $cs \wedge \overline{rhc}$. This means that puc is not available at other preconditions (or constraints).

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_15