Problem set 8 solutions

Planning

Problem 1

Consider a simple blocks world problem:

\[
\begin{array}{c}
B \\
A \\
C & D \\
\end{array}
\quad
\begin{array}{c}
C \\
B \\
A \\
D \\
\end{array}
\]

Initial state Goal state

We use predicates \( \text{On}(x, y) \) and \( \text{Clear}(x) \) to describe the states of the world. \( \text{On}(x, y) \) says that block \( x \) is directly atop block \( y \) and \( \text{Clear}(x) \) says that the top of the block \( x \) is clear.

The initial state is:

\[
\text{On}(B, A), \text{On}(A, C), \text{On}(C, \text{Table}), \text{On}(D, \text{Table}), \text{Clear}(B), \text{Clear}(D).
\]

The goal condition is:

\[
\text{On}(C, B), \text{On}(B, A), \text{On}(A, D).
\]

Part a. Write two STRIPS operators that apply to the blocks world:

- \text{put-on}(x, y) \) for stacking a block \( x \) on another block \( y \), and
- \text{put-table}(x) \) to put the block on the table.
Solution:

**put-on**(x,y):
Precondition: Clear(x), Clear(y) On(x,z), Add: On(x,y),Clear(z) Delete: Clear(y), On(x,z)

**put-table**(x):
Precondition: Clear(x), On(x,z) Add: On(x,Table), Clear(z) Delete: On(x,z)

Remark. To be perfectly consistent, one should make sure that Table does not interfere with our moving and stacking. After all, to the planner Table is just another constant. Right now, just a minor conflict results if, say, put-on(A,Table) is applied. Then, On(A, Table) is added and deleted at the same time and the result might depend on implementation, which is not a desirable thing. But if we decide to add other operations, we might risk inconsistencies. Therefore, the operators should be restricted to operate only on blocks. This can be accomplished by adding Block(A) thru Block(D) to the fact base and using Block(x) in precondition of both **put-on**(x,y) and **put-table**(x).

**Part b.** Assume we search for the plan using forward (goal-progression) search. Describe the state we obtain after the operator **put-table**(B) is applied to the initial state.

**Solution:** The state obtained is:

On(A,C), On(B,Table), On(C,Table), On(D,Table), Clear(A), Clear(B), Clear(D)

**Part c.** An alternative to the forward search is the backward or goal regression search. Assume the operator to be applied just before the goal state is reached is **put-on**(C,B). Describe the new goal that results from the selection of this operator.

**Solution:** Clear(B), Clear(C), On(B, A), On(A, D), On(C, z)

**Part d.** We have decided to use forward search to solve the planning problem. What uninformed method would you consider to solve the problem? Justify.

**Solutions:** The breadth-first search (BFS) method or Iterative deepening (IDA). BFS is complete and optimal. The depth-first search (DFS) can get stuck in cycles or reach suboptimal solution plans. IDA combines the advantages of the BFS and DFS in terms of time and memory complexity while preserving the optimality of the solution plan.

**Part e.** Assume we want to enhance the forward search with a heuristic. To estimate the remaining length of the plan we consider the number of subgoals that remain to be satisfied from the current state. We incorporate the heuristic through an evaluation-function driven search procedure. The evaluation function for the state is \( f(s) = g(s) + h(s) \), where \( g(s) \) is the length of the path from the initial state and the heuristic is the number of subgoals of the goal state that remain to be satisfied by the current state. Draw the search tree generated by the search procedure. Is the solution found optimal?
Solution: The search tree generated during the heuristic search is shown in Figure 1. The particular solution is optimal.

Part f. We are thinking about adopting the number of subgoals heuristic to an arbitrary STRIPS planning problem. Is the evaluation-function search with \( f(s) = g(s) + h(s) \) in combination with such a heuristic always optimal? Explain your answer.

Answer: The solution is not always optimal. The problem with the heuristic is that one operator can satisfy two or more goals in one step so the number of goals to be satisfied can overestimate the length of the solution - hence the heuristic is not admissible. As an example of this consider the following instance of our problem:

\( \text{On}(A, B), \text{On}(C, \text{Table}), \text{On}(B, \text{Table}), \text{Clear}(A), \text{Clear}(C) \)

with the goal:

\( \text{On}(A, C), \text{Clear}(B) \).

Problem 2

Consider a robot whose operation is described by the following STRIPS operators:
• **Action**: Go($x, y$), **Precondition**: At(Robot, $x$), **Add**: At(Robot, $y$), **Delete**: At(Robot, $x$),

• **Action**: Pick($o$), **Precondition**: At(Robot, $x$) $\land$ At($o$, $x$), **Add**: Holding(Robot, $o$),
  **Delete**: At($o$, $x$)

• **Action**: Drop($o$), **Precondition**: At(Robot, $x$) $\land$ Holding(Robot, $o$), **Add**: At($o$, $x$),
  **Delete**: Holding(Robot, $o$)

Assume the initial state is described as:
At(Apple, Room1) $\land$ At(Orange, Room1) $\land$ At(Robot, Room1)
and the goal state is:
At(Apple, Room2) $\land$ At(Orange, Room2).

**Part a.** Draw a complete partial-order plan the POP algorithm would find. Note that there can be more complete partial order plans that are consistent with the problem. You are asked to give only one of these complete plans. Show clearly all causal and ordering links between operators. Give a list of all threats resolved through ordering.

**Solution** solution is shown in Figure 2

**Part b.** List all plans consistent with your partial order plan. **Answer**: There are 4 plans that can be constructed from POP:

1. (a) Pick(Apple)
   (b) Pick(Orange)
   (c) Go(Room1, Room2)
   (d) Drop(Apple)
   (e) Drop(Orange)

2. (a) Pick(Apple)
   (b) Pick(Orange)
   (c) Go(Room1, Room2)
   (d) Drop(Orange)
   (e) Drop(Apple)

3. (a) Pick(Orange)
   (b) Pick(Apple)
   (c) Go(Room1, Room2)
   (d) Drop(Apple)
   (e) Drop(Orange)

4. (a) Pick(Orange)
Figure 2: Complete Partial Order Plan
(b) Pick(Apple)
(c) Go(Room1, Room2)
(d) Drop(Orange)
(e) Drop(Apple)