10. Classical Planning

10.1 Consider a robot whose operation is described by the following PDDL operators:

$$Op(Go(x,y), At(Robot,x), \neg At(Robot,x) \wedge At(Robot,y))$$
$$Op(Pick(o), At(Robot,x) \wedge At(o,x), \neg At(o,x) \wedge Holding(o))$$
$$Op(Drop(o), At(Robot,x) \wedge Holding(o), At(o,x) \wedge \neg Holding(o))$$

1. The operators allow the robot to hold more than one object. Show how to modify them with an $EmptyHand$ predicate for a robot that can hold only one object.
2. Assuming that these are the only actions in the world, write a successor-state axiom for $EmptyHand$.

10.2 Describe the differences and similarities between problem solving and planning.

10.3 [strips-airport-exercise]Given the action schemas and initial state from Figure airport-pddl-algorithm, what are all the applicable concrete instances of $Fly(p, from, to)$ in the state described by

$$At(P_1, JFK) \wedge At(P_2, SFO) \wedge Plane(P_1) \wedge Plane(P_2) \wedge Airport(JFK) \wedge Airport(SFO)?$$

10.4 The monkey-and-bananas problem is faced by a monkey in a laboratory with some bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at A, the bananas at B, and the box at C. The monkey and box have height Low, but if the monkey climbs onto the box he will have height $High$, the same as the bananas. The actions available to the monkey include Go from one place to another, $Push$ an object from one place to another, $ClimbUp$ onto or $ClimbDown$ from an object, and $Grasp$ or $Ungrasp$ an object. The result of a $Grasp$ is that the monkey holds the object if the monkey and object are in the same place at the same height.

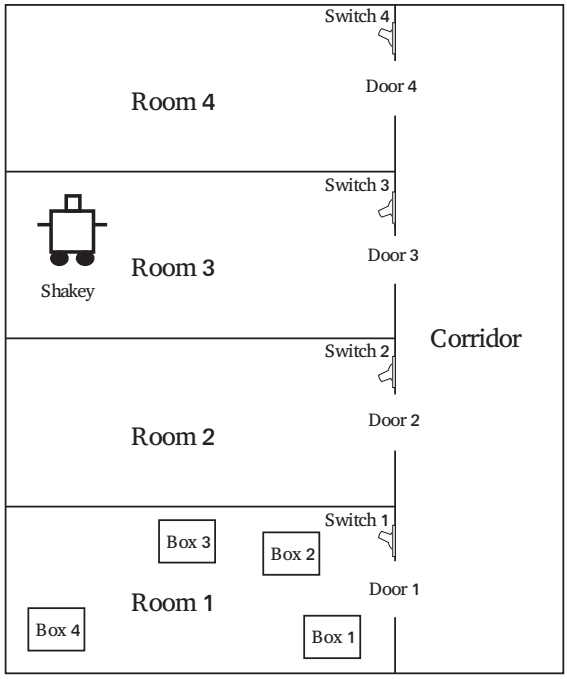
1. Write down the initial state description.
2. Write the six action schemas.
3. Suppose the monkey wants to fool the scientists, who are off to tea, by grabbing the bananas, but leaving the box in its original place. Write this as a general goal (i.e., not assuming that the box is necessarily at C) in the language of situation calculus. Can this goal be solved by a classical planning system?
4. Your schema for pushing is probably incorrect, because if the object is too heavy, its position will remain the same when the $Push$ schema is applied. Fix your action schema to account for heavy objects.

10.5 The original {Strips} planner was designed to control Shakey the robot. Figure shakey-figure shows a version of Shakey's world consisting of four rooms lined up along a corridor, where each room has a door and a light switch. The actions in Shakey's world include moving from place to place, pushing movable objects (such as boxes), climbing onto and down from rigid objects (such as boxes), and turning light switches on and off. The robot itself could not climb on a box or toggle a switch, but the planner was capable of finding and printing out plans that were beyond the robot's abilities. Shakey's six actions are the following:

- $Go(x, y, r)$, which requires that Shakey be $At\ x$ and that x and y are locations In the same room r. By convention a door between two rooms is in both of them.
- Push a box b from location x to location y within the same room: $Push(b, x, y, r)$. You will need the predicate Box and constants for the boxes.
- Climb onto a box from position x: $ClimbUp(x, b)$; climb down from a box to position x: $ClimbDown(b, x)$. We will need the predicate On and the constant $Floor$.
- Turn a light switch on or off: $TurnOn(s, b)$; $TurnOff(s, b)$. To turn a light on or off, Shakey must be on top of a box at the light switch's location.

Write PDDL sentences for Shakey's six actions and the initial state from Figure shakey-figure. Construct a plan for Shakey to get Box_2 into $Room_2$.

Figure [shakey-figure] Shakey's world. Shakey can move between landmarks within a room, can pass through the door between rooms, can climb climbable objects and push pushable objects, and can flip light switches.



10.6 A finite Turing machine has a finite one-dimensional tape of cells, each cell containing one of a finite number of symbols. One cell has a read and write head above it. There is a finite set of states the machine can be in, one of which is the accept state. At each time step, depending on the symbol on the cell under the head and the machine's current state, there are a set of actions we can choose from. Each action involves writing a symbol to the cell under the head, transitioning the machine to a state, and optionally moving the head left or right. The mapping that determines which actions are allowed is the Turing machine's program. Your goal is to control the machine into the accept state.

Represent the Turing machine acceptance problem as a planning problem. If you can do this, it demonstrates that determining whether a planning problem has a solution is at least as hard as the Turing acceptance problem, which is PSPACE-hard.

10.7 [negative-effects-exercise]Explain why dropping negative effects from every action schema results in a relaxed problem, provided that preconditions and goals contain only positive literals.

10.8 [sussman-anomaly-exercise]Figure sussman-anomaly-figure (page sussman-anomaly-figure) shows a blocks-world problem that is known as the {Sussman anomaly}. The problem was considered anomalous because the noninterleaved planners of the early 1970s could not solve it. Write a definition of the problem and solve it, either by hand or with a planning program. A noninterleaved planner is a planner that, when given two subgoals G_1 and G_2, produces either a plan for G_1 concatenated with a plan for G_2, or vice versa. Can a noninterleaved planner solve this problem? How, or why not?

10.9 Prove that backward search with PDDL problems is complete.

10.10 Construct levels 0, 1, and 2 of the planning graph for the problem in Figure airport-pddl-algorithm.

10.11 [graphplan-proof-exercise] Prove the following assertions about planning graphs:

1. A literal that does not appear in the final level of the graph cannot be achieved.
2. The level cost of a literal in a serial graph is no greater than the actual cost of an optimal plan for achieving it.

10.12 We saw that planning graphs can handle only propositional actions. What if we want to use planning graphs for a problem with variables in the goal, such as $At(P_1, x) \wedge At(P_2, x)$, where x is assumed to be bound by an existential quantifier that ranges over a finite domain of locations? How could you encode such a problem to work with planning graphs?

10.13 The set-level heuristic (see page set-level-page) uses a planning graph to estimate the cost of achieving a conjunctive goal from the current state. What relaxed problem is the set-level heuristic the solution to?

10.14 Examine the definition of **bidirectional search** in Chapter search-chapter.

1. Would bidirectional state-space search be a good idea for planning?
2. What about bidirectional search in the space of partial-order plans?
3. Devise a version of partial-order planning in which an action can be added to a plan if its preconditions can be achieved by the effects of actions already in the plan. Explain how to deal with conflicts and ordering constraints. Is the algorithm essentially identical to forward state-space search?

10.15 We contrasted forward and backward state-space searchers with partial-order planners, saying that the latter is a plan-space searcher. Explain how forward and backward state-space search can also be considered plan-space searchers, and say what the plan refinement operators are.

10.16 [satplan-preconditions-exercise] Up to now we have assumed that the plans we create always make sure that an action's preconditions are satisfied. Let us now investigate what propositional successor-state axioms such as $HaveArrow^{t+1} \Leftrightarrow (HaveArrow^t \wedge \neg Shoot^t)$ have to say about actions whose preconditions are not satisfied.

1. Show that the axioms predict that nothing will happen when an action is executed in a state where its preconditions are not satisfied.
2. Consider a plan p that contains the actions required to achieve a goal but also includes illegal actions. Is it the case that $initialstate \wedge successor - stateaxioms \wedge p \models$
3. With first-order successor-state axioms in situation calculus, is it possible to prove that a plan containing illegal actions will achieve the goal?

10.17 [strips-translation-exercise]Consider how to translate a set of action schemas into the successor-state axioms of situation calculus.

1. Consider the schema for $Fly(p, from, to)$. Write a logical definition for the predicate $Poss(Fly(p, from, to), s)$, which is true if the preconditions for $Fly(p, from, to)$ are satisfied in situation s.
2. Next, assuming that $Fly(p, from, to)$ is the only action schema available to the agent, write down a successor-state axiom for $At(p, x, s)$ that captures the same information as the action schema.
3. Now suppose there is an additional method of travel: $Teleport(p, from, to)$. It has the additional precondition $\neg Warped(p)$ and the additional effect $Warped(p)$. Explain how the situation calculus knowledge base must be modified.
4. Finally, develop a general and precisely specified procedure for carrying out the translation from a set of action schemas to a set of successor-state axioms.

10.18 [disjunctive-satplan-exercise] In the {SATPlan} algorithm in Figure satplan-agent-algorithm (page satplan-agent-algorithm), each call to the satisfiability algorithm asserts a goal g^T, where T ranges from 0 to $T_{\max}$. Suppose instead that the satisfiability algorithm is called only once, with the goal $g^0 \vee g^1 \vee \cdots \vee g^{T_{\max}}$.

1. Will this always return a plan if one exists with length less than or equal to $T_{\max}$?
2. Does this approach introduce any new spurious “solutions”?
3. Discuss how one might modify a satisfiability algorithm such as {WalkSAT} so that it finds short solutions (if they exist) when given a disjunctive goal of this form.