6.825 Midterm Exam Fall 2002

October 29, 2002

1. (8 points) Convert the following four sentences to clausal form:

   (a) $\forall x.\exists y.p(x) \wedge r(x,y)$
   Solution:

   $$p(x_1)$$
   $$r(x_2, F(x_2))$$

   (b) $\exists x.\forall y.p(x) \rightarrow r(x,y)$
   Solution:

   $$\neg p(\mathrm{A}) \vee r(\mathrm{A}, y_1)$$

   (c) $\forall x.(\exists y.r(x,y)) \leftrightarrow p(x)$
   Solution:

   $$\neg r(x_1, y_1) \vee p(x_1)$$
   $$\neg p(x_2) \vee r(x_2, F(x_2))$$

   (d) $\neg\forall x.\exists y.p(x) \wedge r(x,y)$
   Solution:

   $$\neg p(\mathrm{A}) \vee \neg r(\mathrm{A}, y_1)$$

2. (8 points) For each group of sentences below, give an interpretation that makes the first sentence(s) true and the last sentence false. Use $\{A, B, C\}$ as your universe.

   (a)

$$\exists x.p(x) \wedge q(x) \wedge r(x,x)$$
$$\forall x.p(x) \rightarrow \exists y.\neg r(x,y)$$
$$\forall x.p(x) \rightarrow \exists y.\neg x = y \wedge r(x,y)$$

$$\forall x.p(x) \vee \neg q(x)$$

   Solution:

$$\begin{aligned} p &= \{< A >\} \\ q &= \{< A, C >\} \\ r &= \{< A, A >, < A, B >\} \end{aligned}$$

   (b)

$$\forall x.p(x) \leftrightarrow \exists y.r(y,x)$$
$$\forall x.\exists y.r(x,y)$$

$$\forall x.\neg p(x)$$

   Solution:

$$\begin{aligned} p &= \{< A >\} \\ r &= \{< A, A >, < B, A >, < C, A >\} \end{aligned}$$

   Alternately: (there are others, too)

$$\begin{aligned} p &= \{< A, B, C >\} \\ r &= \{< A, A >, < B, B >, < C, C >\} \end{aligned}$$

3. (8 points) True or False?

   (a) The resolution/refutation proof strategy for propositional logic runs in polynomial time in the size of the input.

   Solution:

   False. The result of converting to clausal form alone can be exponentially larger than the input.

   (b) GraphPlan runs in time polynomial in plan depth.

   Solution:

   False. While creating the plan graph is polynomial, searching for a plan is still non-polynomial.

   (c) Iterative deepening requires less space than breadth-first search.

   Solution:

   True. Iterative deepening uses only as much space as depth-first search.

   (d) In first-order logic, if a formula is entailed by a theory, it can always be proven using the resolution/refutation proof strategy.

   Solution:

   True, resolution-refutation is a complete proof procedure for first-order logic.

4. (30 points) A robot has to deliver identical packages to locations A, B, and C, in an office environment. Assume it starts off holding all three packages. The environment is represented as a grid of squares, some of which are free (so the robot can move into them) and some of which are occupied (by walls, doors, etc.). The robot can move into neighboring squares, and can pick up and drop packages if they are in the same square as the robot.

   (a) (4 points) Formulate this problem as a search problem, specifying the state space, action space, goal test, and cost function.

   Solution:

   The state space needs to include enough information so that, by looking at the current values of the state features, the robot knows what it needs to do. For this task, the robot needs to be able to determine its position in the grid and which packages it has already delivered.

   - State: { (x,y), deliveredA, deliveredB, deliveredC }
   - Action space: MoveN, MoveE, MoveS, MoveW, DropA, DropB, DropC.
   - Goal test: { (x,y), delA, delB, delC } = { (any x, any y), 1, 1, 1 }
   - Cost function: cost of 1 for each action taken.

Common Mistakes:

- Including only the number of packages in the state space: this is insufficient because, if we know we are holding 2 packages, we don't know if, for example, we've already been to A, or if we should try to go there next.
- Specifying a goal test that couldn't be derived from the state space that was described.

(b) (3 points) Give a non-trivial, polynomial-time, admissible heuristic for this domain, or argue that there isn't one.

Solution:

There were many acceptable solutions. Some examples:

- Straight line or manhattan distance to furthest unvisited drop-off location.
- Straight line or manhattan distance to nearest unvisited drop-off location. (slightly looser underestimate than the previous one)
- Number of packages left to deliver. (Not as "good" as the above two, but still acceptable.)

Common Mistakes:

- Trying to solve the Travelling Salesman Problem: In the general case, not poly in the number of drop-off locations.
- Summing the distances between current robot location and all unvisited drop-off locations: Can overestimate the distance.
- Summing up the perimeter of the convex hull of robot location and drop-off locations: Can overestimate the distance.

(c) (3 points) Package A has to be delivered to the boss, and it's important that it be done quickly. But we should deliver the other packages promptly as well. How could we encode this in the problem?

Solution:

The idea of putting "pressure" on the delivering to the boss is correctly handled in the cost function. We want to emphasize delivery to A, but we don't want to ignore B if we happen to pass by it. Please note that to encode something into the "problem," it must be encoded into one of the problem components: the state space, action space, goal test, or cost function. For example:

while we haven't delivered to A, actions cost 2; after that, actions cost 1.

Common Mistakes:

- Decreasing the cost instead of increasing the cost.
- Constraining the order of the problem; requiring delivery to A before considering the other two locations at all.
- Modifying the search heuristic instead of the cost function.

(d) (3 points) Now, consider the case where the robot doesn't start with the packages, but it has to pick up a package from location 1 to deliver to location A, a package from location 2 to deliver to B, and from 3 to deliver to C. What is an appropriate state space for this problem?

Solution:

Again, multiple solutions were possible. One example:

- State: { (x,y), deliveredA, deliveredB deliveredC, holdingA, holdingB, holdingC }

Common Mistakes:

- Again, just the number of packages and the number of visited locations are insufficient.

(e) (3 points) What is a good admissible heuristic?

Solution:

- The same heuristic as before. (Not so "good," but acceptable.)
- The maximum (distance from my location to unvisited pickup location s_i plus the distance from s_i to unvisited delivery location).

(f) (4 points) This problem can also be treated as a planning problem.

Describe the effects of the action of moving north using situation calculus axioms. Assume that we are representing the location of the robot using integer coordinates for its x, y location in the grid, and that we can name the next integer coordinate greater than x as $x + 1$. Additionally, assume that we have a predicate *occupied* that applies to x, y pairs, indicating whether the robot may pass through the indicated locations.

Solution:

$$\forall x, y, s. atrobot(x, y, s) \wedge \neg occupied(x, y + 1) \rightarrow atrobot(x, y + 1, result(moveNorth, s))$$

A little variability in this answer was acceptable, but the idea of *result* as a function from actions and situations to actions was crucial.

(g) (3 points) Describe what frame axioms would be necessary for this domain (but don't write them down).

Solution:

- When the robot moves, the positions of the (non-held) packages and the delivery locations don't change; which packages are held doesn't change.
- When the robot picks up or drops off a package, the robot's location doesn't change; the locations of packages and delivery locations don't change.

Some people mentioned that which squares are occupied doesn't change. Because these facts never change (unless we have moving obstacles), it would probably be best to model them without a situation argument, thereby obviating the need for frame axioms for *occupied*.

(h) (3 points) Either provide a description of the move operation in the STRIPS language, or say why it is difficult to do so.

Solution:

There were a couple of different acceptable answers here. The most common one was some variation on :

move(x,y):

Pre: *at(x)*, ¬ *occupied(y)*, *neighbor(x,y)*
Effect: ¬ *at(x)*, *at(y)*

This requires some mention of the fact that it might be a big pain to specify the *neighbor* relation.

Another answer was

moveNorth(x,y):

Pre: *at(x,y)*, ¬ *occupied(x,y+1)*
Effect: ¬ *at(x,y)*, *at(x,y+1)*

This requires some mention, at least, of the fact that you'd need 4 operators. I gave full credit for this, but, in general, it's not legal to use functions in STRIPS, so you can't write "y+1" in the effects.

One more answer, saying that it was too hard, because you'd need to have a specific operator for each possible location (due to the fact that you can't have functions and/or that the "neighbor" relation would be hard to specify) was also okay.

(i) (2 points) When does it make sense to treat a problem as a planning problem rather than as a basic state-space search problem?

Solution:

There are a lot of decent answers here. I was looking for at least a couple of the following points.

- The initial state is not known exactly (so you have to search over the space of sets of states).
- There is a direction connection between actions and effects that can be heuristically exploited.
- The goal is described conjunctively, so solving the subgoals individually might lead to a plan more efficiently.
- The domain is described using a factored, logical representation (this really contributes to all of the above points).

- There is not a big concern with minimizing cost (because planning methods usually seek any path to the goal).

Saying "when the state space is too big" was not sufficient.

(j) (2 points) Should this problem be treated as a planning problem? Explain why or why not.

Solution:

I accepted a lot of answers here, mostly dependent on how cogently they were argued in the context of the answer to the previous problem.

I actually think that treating this as a planning problem won't be much help because: the initial state is known exactly, the subgoals aren't particularly separable (you can't solve the problem of going to location A without knowing where you're going to be when you start), and we have a strong desire for minimizing path cost.