Questions Answers: 1, 2, 3, 5, 6

KBAI CS 7637/4635, OMS/PE CS 7637
Magahet Mendiola
Spring 2015

Mid-Term Examination

Question 1: Specify the STRIPS planning operators for this microworld: Walk, Push_Table, Climb-Up, Climb_Down, Grasp_Wire, Charge. Here is an example:

Climb_Up:
Precondition: Location (Robot, On_Floor)
Postcondition: Location (Robot, Next_to_Wires)

ANSWER 1:

Walk:
Precondition: Location(Robot, x)
Postcondition: Location(Robot, y)

Push_Table:
Precondition: Location(Robot, x) & Location(Table, x)
Postcondition: Location(Robot, y) & Location(Table, y)

Climb_Up:
Precondition: On(p, Floor)
Postcondition: On(p, y)

Climb_Down:
Precondition: On(p, y)
Postcondition: On(p, Floor)

Grasp_Wire:
Precondition: Location(Robot, Center) & On(Robot, Table)
Postcondition: Held(Robot, Wires)

Charge
Precondition: Uncharged(Robot) & Held(Robot, Wires)
Postcondition: Charged(Robot)

Question 2: Now consider a subset of the planning problem: in the initial state the robot is in one corner and the table is in another corner of the room; in the goal state robot is charged (and we don't care about the final location of the robot or the table – the planning stops once the robot is charged).

Show how a STRIPS like planner can form a plan to achieve the above goal.

ANSWER 2:

The problem can be expressed with the following STRIPS description:

Init: Location(Robot, Corner1) & Location(Table, Corner2)
Goal: Charged(Robot)
Actions (Simplified from Question 1 to expedite planning):

Walk:
Precondition: Location(Robot, Corner1)
Postcondition: Location(Robot, Corner2)

Push_Table:
Precondition: Location(Robot, Corner2)
Postcondition: Location(Robot, Center) & Location(Table, Center)

Climb_Up:
Precondition: Location(Robot, Center) & Location(Table, Center)
Postcondition: Location (Robot, Next_to_Wires)

Climb_Down:
Precondition: Location (Robot, Next_to_Wires)
Postcondition: Location(Robot, Center)

Grasp_Wire:
Precondition: Location (Robot, Next_to_Wires)
Postcondition: Held(Robot, Wires)

Charge:
Precondition: Held(Robot, Wires)
Postcondition: Charged(Robot)

From here we can use the partial order planning to create a complete plan to achieve the end goal. Here is the procedure we would follow to create the final plan:

1. Set the initial and goal states.

2. Select an open precondition, in this case there is only one: Charged(Robot)

3. Select an action whose postcondition matches the open precondition: Charge(Robot)

4. Having added the previous action, we now remove the met precondition and add those preconditions from the chosen action: Held(Robot, Wires)

5. Held(Robot, Wires) can be can be achieved with Grasp_Wire(Robot), which we will add to our plan: Init → ___ → Grasp_Wire(Robot) → Charge(Robot) → Goal

6. Again, we remove the met precondition and add preconditions from the added actions. This is the current set of open preconditions: Location (Robot, Next_to_Wires)

7. We then add the action, Climb_Up(Robot, Table). Open preconditions are now: Location(Robot, Center) & Location(Table, Center)

8. Push_Table meets both of these conditions, so we add that to our plan, which is now: Init → ___ → Push_Table(Robot) → Climb_Up → Grasp_Wire(Robot) → Charge(Robot) → Goal

9. Push_Table has Location(Robot, Corner2) as a precondition, which can be met with Walk(Robot, Corner2).

10. Finally, Walk requires Location(Robot, Corner1), which matches the init state. The complete plan is Init → Walk(Robot, Corner2) → Push_Table(Robot, Center) → Climb_Up(Robot, Table) → Grasp_Wire(Robot) → Charge(Robot) → Goal.

Each action is chosen based on the set of preconditions currently unmet until the init goal is met. In this case there was no need to resolve conflicts as each state lent itself to the next. However, with the more complex set of actions available in Question 1, we could have required the coordination of the robot's location, the table's location, and achieving the state, On(Robot, Table).

Question 3: Rule-based reasoning is another method the robot could have used to address this situation. Invent rules for this microworld. Here is an example in English (you want to use propositional logic in your answer):

Rule: If the Electrical wire is too high to reach and a Table is in a corner,
Then Push the Table to beneath the wires and Climb on the Table.

Given the rules such as the one shown above, show how rule-based reasoning would address Q2.

ANSWER 3:

We will setup the following rules:

Rule: Location(Robot, Corner1),
Then Walk(Robot, Corner2) & Set Location(Robot, Corner2)

Rule: Location(Robot, Corner2) & Location(Table, Corner2),
Then Push_Table(Robot, Center) & Set Location(Robot, Center) & Location(Table, Center)

Rule: Location(Robot, Center) & Location(Table, Center),
Then Climb_Table(Robot) & Set Location(Robot, Next_to_Wires)

Rule: Location(Robot, Next_to_Wires),
Then Grasp_Wires(Robot) & Set Held(Robot, Wires)

Rule: Held(Robot, Wires),
Then Charge(Robot) & Set Charged(Robot)

Using these rules, the robot will match against each, in order, and will update the current state as it proceeds. Starting from the initial state of Location(Robot, Corner1), each rule precept will match the current state and perform the given action. By the last rule, the agent will have achieved the goal state.

Question 4: Case-based reasoning is yet another method for explaining the behavior of the robot. Invent the case for addressing Q2. You may build on your answer to Q2 to compile the case. Show its contents and indices.

Now given the above case in robot's memory, show how given the new situation of the full planning problem (the robot is not only charged, but also the robot and the table are in their old corners), the robot retrieves and adapts the old case to address Q2.

Question 5: Suppose that one of the various rules in your answer to Q3 is missing so that the robot cannot use rule-based reasoning. Modify your answer to Q3 to show how this will lead to an impasse.

Now invent a case from which the missing rule can be learned and show how the rule-based reasoner can complete its reasoning.

ANSWER 5:

We start with the same rules as question 3, with the exception of one:

Rule: Location(Robot, Corner1),
Then Walk(Robot, Corner2) & Set Location(Robot, Corner2)

Rule: Location(Robot, Corner2) & Location(Table, Corner2),
Then Push_Table(Robot, Center) & Set Location(Robot, Center) & Location(Table, Center)

Rule: Location(Robot, Center) & Location(Table, Center),
Then Climb_Table(Robot) & Set Location(Robot, Next_to_Wires)

====== We will consider this rule removed ======
Rule: Location(Robot, Next_to_Wires),
Then Grasp_Wires(Robot) & Set Held(Robot, Wires)
==

Rule: Held(Robot, Wires),
Then Charge(Robot) & Set Charged(Robot)

Without the Grasp_Wires rule, the agent would be unable to set Held(Robot, Wires) in working memory. However, if we are provided a case where working memory has achieved this state, we can compare working memory in that case to those other cases which have not reached that state. Our reasoner would see the discrepancy as: No rule exists that changes working memory to included Held(Robot, Wires). The agent would then add the Grasp_Wires rule to long-term memory to allow the next state to be reached and the agent to progress to the goal state.

Question 6: Let us suppose that the robot did not initially have the concept of a Table. However, its previous owner taught the concept by giving some examples one at a time in the order shown here:

Positive Example:	Negative Example:	Positive Example:
(Table (Top Flat)	(Table (Top Flat)	(Table (Top Flat)
(Legs 4)	(Legs 3)	(Legs 4)
(Material Wood))	(Material Wood))	(Material Metal))

Let us suppose that the robot has background knowledge that wood and metal are both solid materials. Invent generalization and specialization heuristics for incremental concept learning this domain.

Now show how the robot will learn the concept of a table from these examples.

ANSWER 6:

The robot begins with an empty concept.

The first example is given, which updates the current table concept as:

(Table(Top Flat)
(Legs 4)
(Material Wood))

Specialization:

The second example shows that one of the parts differs from the current table concept. To allow for specialization, in this case, we need a heuristic that checks for mismatched parts between negative examples and the current concept and changes the current concept part to be more specialized. In this case, (Legs 4) would become (required Legs 4). The updated concept would require exactly 4 legs:

(Table(Top Flat)
(required Legs 4)
(Material Wood))

Generalization:

The third example shows that our current concept is too narrow since it is limited to (Material Wood). Our generalization heuristic would detect that this is a positive example, and would find mismatches between concept parts. It would then determine that there is a current working knowledge that a is-a relationship exists between Wood and Solids as well as between Metal and Solids. The robot would use this knowledge to “climb” the is-a relationship tree and update the table concept with a generalized part (Material Solid):

```
(Table(Top Flat)
        (required Legs 4)
        (Material Solid))
```

Question 7: Now suppose that the robot does not have the knowledge required by the above questions. Thus, it doesn't have the cases or the rules or even the STRIPS planning technique. How might the robot then accomplish the goal of Q2? Illustrate. (Hint: if the robot does not have access to knowledge-based methods, it may resort to general-purpose methods such as Generate and Test.)