Interleaved Planning

1. Recall our overall model:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Information</th>
<th>Attribute</th>
<th>Simple Model</th>
<th>Abstract Model</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>Binary</td>
<td>Detection</td>
<td>Maps</td>
<td>Logic</td>
<td>Agent Modeling</td>
</tr>
<tr>
<td>Motor</td>
<td>Kinematics</td>
<td>Action Selection</td>
<td>Path Planning</td>
<td>Task Planning</td>
<td>Goal Selection</td>
</tr>
</tbody>
</table>

2. We have moved a long way up the layers. Now we can plan paths both in configuration space and in real space in order to get robots in the correct position to manipulate objects.

3. The next step is to figure out what we want to manipulate, and how we want to manipulate that thing.

4. Typically, this comes from some task, some complex goal that we want the robot to solve.
   (a) Take a complex task you may have discussed in your introduction to programming course—making a sandwich.
   (b) Assume our robot knows about making sandwiches in general and we ask it to make us a grilled lobster and cheese.
   (c) The robot knows the current state of the world, piles of bread, lobster and cheese slices, and now knows the goal (a grilled lobster and cheese sandwich). It needs to figure out what intermediate steps need to be done.
   (d) This is an example of a classic AI planning problem, and we will apply classic AI planning techniques, but there is one problem to keep in mind when I’m talking about this: just because a robot tries to do something, doesn’t mean it gets done.
   (e) This means that even is we have a plan, we need to watch it for failure and possibly re-plan.

5. This gives us the Sense → Plan → Act loop.
   (a) Gather information from all the sensors.
   (b) Integrate this information together to build a world model.
   (c) Define a search space based on this model.
   (d) Formulate a plan of action to reach a goal
(e) Decide on the first action to take to execute that plan.

(f) Feed this into the path planner, which can find a path, and feed that down to the action selection module.

(g) Incorporate the path into the set of survival goals and select an action.

(h) Calculate the kinematics necessary to take that action

(i) Take it.

(j) Repeat.

6. So let us look at what it takes to do all this.

(a) Sensing. We’ve talked a lot about sensing, but the basic idea is to identify properties in the environment. We can do this with the sensor fusion from the detection layer, from camera and video based techniques (which would have to be covered in a separate course) and from the maps we built.

(b) World model. At this level the world model is abstract statements about the world, as opposed the direct representations of the would, as in a map. Our statements need to follow consistent rules for their expression and reasoning about them. That is the definition of a logic. We need to represent this as a logic.

(c) Once we have the world (and our goal) represented in some logic, we need to figure out the sequence of actions that gets us from now to then. That is the planning step.

   i. Note: we will make what is known as the Closed World Assumption. This assumption says that anything we don’t explicitly state as true is assumed to be false. If we don’t know some facts, then the rules of logic don’t allow us to come to conclusions. There are techniques for dealing with this, and they’re known as probability.

   ii. From here on we’ll look at a particular environment as an example: block. Take this situation:

   \[
   \begin{array}{c}
   \text{B} \\
   \text{A} \\
   \text{D} \\
   \text{C} \\
   \text{Fl1} & \text{Fl2} & \text{Fl3}
   \end{array}
   \]

   iii. This can be described with the following 5 logical statements:

   A. On(B,A)
   B. On(A,D)
   C. On(D,C)
   D. On(C,Fl1)
   E. Clear(B)
   F. Clear(Fl2)
   G. Clear(Fl3)
iv. This can be thought of as the state of the world. The goal state could be:
   A. On(A,B)
   B. On(B,C)
   C. On(C,D)
   D. On(D,Fl1)
   E. Clear(A)
   F. Clear(Fl2)
   G. Clear(Fl3)

v. The robot can also perform actions in the environment:
   A. move(x,y,z)- move block x from y to z.

vi. We can take world states and connect them with links.

vii. A link goes from one state to another if the robot can transform the world from the one state to the other in a single action.

viii. The result is a (likely infinite) graph.

(d) The robot will then repeatedly apply operators to states, generating new states until the robot can generate the goal state.
   i. The ordered set of operators that transforms the current state into the goal state is the plan.
   ii. To the graph, we can apply any of the standard search algorithms:
       A. DFS,
       B. BFS,
       C. A*, etc.

(e) Once the robot has a plan, it selects the first action from the set as the one to execute.

(f) The robot then has an action to execute in the form of: moveTo(pose). This can be fed to the path planner.

(g) and the rest, you already know. It feeds back out the output channel to the motors.

7. So how do we build the graph? First note that we don't build the graph explicitly. The graph is possibly infinite, and if finite, unlikely to fit in memory. Instead we generate adjacent vertices as we do the search.

   (a) We generate vertices by applying operators. These are rules that tell us what the neighboring states are, and what properties they have.
   (b) Each action the robot can take has a rule for determining the neighboring states.
   (c) The basic idea is that the rule (called an operator) looks at the set of logic statements that describe the world state, looking for clauses that make the operator applicable. For example, in order to drop something, the robot needs to be holding something.
   (d) The rest of the rule describes what changes are made to the world state in the neighboring state. So if the robot drops a hammer, it is no longer holding a hammer (that clause is deleted) and the hammer is now on the floor (a new clause is created).
(e) So our operators take the form of three lists: preconditions that determine if the rule can be used, list of clauses to be deleted from the world state.

(f) Take, move\((x,y,z)\) (move x from y to z):

i. preconditions: \(On(x,y) \land Clear(x) \land Clear(z)\).

ii. delete: \(On(x,y), Clear(z)\)

iii. add: \(On(x,z), Clear(y)\)

8. So lets run through a serious mobile robot example, the room example.

(a) Predicates:

i. INROOM\((x,r)\) - x is an object, r is room.

ii. NEXTTO\((x,t)\) - x is an object, t is an object.

iii. STATUS\((d,s)\) - d is a door, s is a status, OPEN or CLOSED.

iv. CONNECTS\((d,rx,ry)\) - d is a door, rx and ry are rooms.

v. HOLDING\((r,x)\) - r is a robot, x is a liftable object

vi. LIFTABLE\((x)\) - x is an object

(b) Imagine an initial state:

i. INROOM\((IT,R1)\) - IT is the name of our robot.

ii. INROOM\((Book1,R2)\)

iii. INROOM\((TABLE1,R1)\)

iv. INROOM\((PLEBE1,R1)\)

v. CONNECTS\((D1, R1,R2)\)

vi. CONNECTS\((D1, R2,R1)\) - really, there is a reason for this.

vii. STATUS\((D1,CLOSED)\).

viii. NEXTTO\((IT,TABLE1)\)

(c) And a goal state:

i. HOLDING\((IT,BOOK1)\).

ii. STATUS\((D1,CLOSED)\).

(d) Operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Preconditions</th>
<th>Add-list</th>
<th>Delete-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOTO((IT,x,y))</td>
<td>INROOM((IT,r))\wedge INROOM((y,r))\wedge NEXTTO((IT,x))</td>
<td>NEXTTO((IT,y))</td>
<td>NEXTTO((IT,x))</td>
</tr>
<tr>
<td>OPEN((IT,d))</td>
<td>NEXTTO((IT,d))\wedge STATUS((d,CLOSED))</td>
<td>STATUS((d,OPEN))</td>
<td>STATUS((d,CLOSED))</td>
</tr>
<tr>
<td>GOTHRU((IT,d))</td>
<td>INROOM((IT,rx))\wedge NEXTTO((IT,d))\wedge CONNECTS((d,rx,ry))\wedge STATUS((d,OPEN))</td>
<td>INROOM((IT,ry))</td>
<td>INROOM((IT,rx))</td>
</tr>
<tr>
<td>CLOSE((IT,d))</td>
<td>NEXTTO((IT,d))\wedge STATUS((d,CLOSED))</td>
<td>STATUS((d,OPEN))</td>
<td>STATUS((d,OPEN))</td>
</tr>
<tr>
<td>PICKUP((IT,x))</td>
<td>NEXTTO((IT,x))\wedge LIFTABLE((x))</td>
<td>HOLDING((IT,x))</td>
<td>LIFTABLE((x))</td>
</tr>
</tbody>
</table>

9. Some things to think about:

   (a) Closed world assumption says we only plan across the things we know about.
(b) There can be no surprises, but, what if we forget a detail that is needed for some particular task?
(c) How many facts would there need to be to cover a situation that is reasonably complicated?
(d) How big is the search space and how much time will we spend?
(e) What about all the irrelevant details that we still have to reason over?
(f) We can break it into pieces, but these have little portability.