RI16x62: Lab 6
Programming Architectures

Due: Tuesday, Week 10

Introduction:

A programming architecture encompasses the planning aspects of a problem as well as the execution of the plan. With the planners you created in the last lab, the programming architecture is simple – just `execute` the plan and the robot achieves the goal (assuming your `GTNN` and `TurnTo`s work perfectly). There are, however, more interesting and complex programming architectures, two of which you will develop for this lab. In the first assignment, the assumptive programming architecture "assumes" it has perfect information about the environment even though it does not. In the second assignment, you will create a delayed planning architecture, which enables the robot to interleave multiple episodes of planning and execution.

Assignment 6.0: Assumptive Programming Architecture

By adding simple logic to a sequential planner, you can deal with some simple cases of incomplete information. Assume that your robot still has complete knowledge about its position and orientation in the world. However, your map is now incomplete. That is, the given map specifies NONE of the walls in the real world (except the walls around the perimeter of the rectangle). In other words, every wall in the map is in the real world, but not every wall in the world is in the map.

The robot could temporarily assume one of the states in its state set is the actual state of the world. For instance, the robot could assume that there are no walls other than the walls it has already seen! Then, the robot can plan under this false pretense. During execution, the robot can compare its percepts (`WhatDoISee`) to what it expects to see according to its map. If it recognizes a discrepancy, it can minimally modify its map to resolve the discrepancy and re-plan from its new position, if necessary. This is one form of assumptive programming, in which you make simplifying assumptions about the structure of the maze's internal walls.

Implement an assumptive planning architecture on your robot. The robot should use assumptive programming to attempt to reach a state in `the specified goals`. The robot may assume that all walls specified in the `mazeworld` do exist; however, there *will* be other walls in the maze that were not specified. If there is a path to a goal, the robot must reach a goal. If there is no path, the robot must indicate this after discovering this fact. The planner component of your assumptive system need only search to depth `<max-len>` as before (given in the file).

We will test your code by placing it in a world, probably a 3 x 6 world, and so specify in the file, indicating all walls around the perimeter only. We will make the goal set contain more than one goal but we guarantee that the initial state set will contain only a single state (robot position) and we will ask your robot to achieve one using assumptive interleaving.

Assignment 6.1: Delayed Planning Architecture: Interleaving Planning and Execution!

Robots often have incomplete information about the state of the world. When the robot's uncertainty is a hindrance to goal-satisfaction, then sequential planners can fail altogether. When the robot's percepts provide feedback that helps resolve the uncertainty, then conditional planning can save the day by providing traces of execution that depend, or condition, on run-time perceptual inputs.

Unfortunately, conditional planning time is a serious drawback. If we triple the size of the environment and have your robot attempt to figure out where it is, we would have to wait several hundred years for your planner to find the huge but optimal conditional solution. Of course, the execution time would be a few minutes. How can we compromise the ideal nature of our execution time to bring the total time below 100years+several minutes?

The solution lies in interleaving planning and execution. If we can come up with conditional plans that do not take the robot to the goal – but provide some progress nevertheless – then execute these plans, we can use the perceptual feedback during execution to narrow down our set of possible world states (this sentence provides two hefty hints). By interleaving planning and execution, we mean precisely this: repeatedly creating a partial plan then executing this partial plan and reaching a state of mind with more complete information (fewer possible world states).

Implement the delayed planning architecture that handles uncertainty about the robot's positions. As with labs 5 and labs 4, you can assume complete knowledge about the maze map. Furthermore, your interleaving system must reach the goal but does *not* need to be optimal.

Hints Do a good job.