Lab 4: Planning

EECS 348

February 27, 2018

In this lab assignment, you are going to define a domain for solving given problems. The domain and the problems are defined using Planning Domain Definition Language (PDDL). In defining the domain, you will need to determine the appropriate types predicates and actions to find plans for each of the problems.

1 Starter code and data

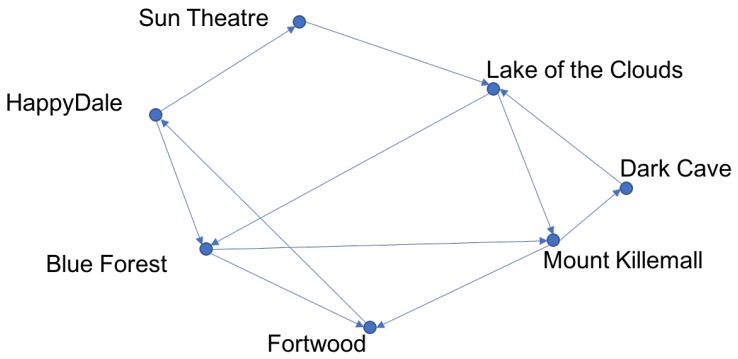
We provide you two files with code: 'main.py' and 'planner.py'. (Details about these files are described at the end of this write-up.)

- 'main.py' contains code for running the planner and testing the output
- 'planner.py' contains function for calling a planner (which is running on a remote server)

There are also six data files: `student_domain.pddl`, `task00.pddl`, `task01.pddl`, `task02.pddl`, `task03.pddl`, and `task04.pddl`. The `student_domain.pddl` file contains an outline for defining the domain. This file is where you will be adding content. The other data files contain definitions of similar problems, and each will be used in a test of planning for the domain you have defined.

2 The domain

You will be defining the domain of our AI hero saving the town of Happy Dale from the evil dragon Nosliw. The dragon is in the Dark Cave in Mount Killemall. Other relevant locations are the Sun Theatre, Lake of the Clouds, Blue Forest, and Fortwood. There are paths between many of these locations, but the paths can only be traversed in one direction. The map below indicats the locations and the paths:



In addition to our AI hero and the dragon, there are a few other characters. There is a bard at the Sun Theatre, Sarorah the Sorceress in the Blue Forest, and the wizard Whitebeard at the Lake of the Clouds.

Each agent is willing to trade items. Additionally, any wizard (including a sorceress) is willing to trade 3 diamonds for magical strength.

Lastly, there are two ways to save the town of Happy Dale. First, the hero needs to be strong and then go to the Dark Cave to slay the evil dragon. Alternatively, the hero can put the dragon into a deep, deep sleep. To accomplish this, the hero needs to get a quill and then write a song that puts the dragon to sleep.

3 Your task

Your task is to define the types, predicates, and actions of the domain so that plans for the given problems may be found. To get you started, an outline of the necessary components are given in the `student_domain.pddl` file. This include sections for `types` and `predicates`. Also included is a template for defining an action.

Begin by defining the types and predicates that you will need. The task files contain all of the types and predicates you will need to define. In defining types, you will want to create a type hierarchy. For example, below we have defined an object type and three other types that are objects:

```
item agent location - object
```

Each of the types on the left can then be used to define further subtyping.

In defining the predicates, use the types to constrain each argument to the predicate, but be sure to not too narrowly define the types.

Lastly, you will define the actions necessary to plan for each problem. Begin with task00, which involves going from one location to another location and then obtaining an item once there. Define actions to have our hero move from Happy Dale to a nearby location, as long as there is a path from Happy Dale to that location.

4 Testing

To test this assignment, we will create several more problems similar to the problems given. For many of these problems, the domain you define should allow for a plan to be found to solve the problem. In other problems, no plan should be found. Note that it is possible to define trivial domains that will allow the hero to save the town in a single step. These sorts of solutions will not generalize to other problems and will fail on tests. Also note that the solver is not guaranteed to find an optimal path. Your solution may involve unnecessary actions such as extra trades. If your path solves the problem correctly and is within the bounds don't worry about it being an optimal solution.

Examples of other possible problems include the following:

1. invalid use of predicates, like having a goal similar to (at suntheatre happydale)
2. trading for an item when having no item to give
3. going from Happy Dale to the Sun Theatre and back to Happy Dale in just 2 steps
4. killing the dragon without being strong
5. putting the dragon to sleep without having the quill

You are encouraged to create new problem files to test for each of these scenarios and more.

5 Hints

Travel: Start by defining how the hero (and only the hero) would move from one location to another. Be sure to properly define the preconditions, checking that there is a path. Also, when traveling from A to B, one has to start by being at A.

Trade: Exchanging goods can be easily done, provided that the two characters are at the location and each has an item to trade. Do not worry about constraining which items or types of items anyone would trade for. In other words, anyone will trade for anything.

Pickup: If the hero finds an object on the ground (e.g., `(at d3 mtkillemall)`), then the hero can pick it up.

Drop: Converse to picking an object up, the hero can choose to drop an item on the ground at any time, thus making that object at that location (e.g., `(at pointy mtkillemall)`).

Magic: Any wizard will give the hero magical strength in exchange for 3 diamonds.

Song: Once the hero has obtained a quill, the hero can write a song putting the dragon into a deep sleep. One the dragon is fast asleep, the town of Happy Dale is safe.

Slaying: Once the hero is strong, the hero may use a sword to slay the dragon. The hero must be at the same location as the dragon in order to slay it. One the dragon is dead, the town of Happy Dale is safe.

6 Files

main.py: Each test calls `planner.find_plan` with the name of the domain and problem files. The returned `Plan` is tested to determine if a plan was found, how long is the plan, the contents of the plan, etc.

planner.py: This file is responsible for connecting to a remote server to find a plan for the given problem and domain. Once the remote server returns, the results are stored in a Plan object. If planning is successful, then the `ok` field will be true, the `plan` field will have a list of the names of the actions in the plan, and `actions` will have the full description of each action in the plan In the event that the planning was not successful, `ok` will be `False` and `error` will contain the error message.

7 Resources

In `planner.py`, we use a remote server to solve the plans. You can try out this server directly. In particular, go to `editor.planner.domains` in your web browser. There, you can load many examples of PDDL. It also provides an interface for editing your PDDL files and running the planner (solver) directly.

8 Extra

All of the problems should be solvable even if you constrain the hero to possessing no more than 3 items. As an extra challenge, you can define your actions to enforce this constraint. No extra credit is given for this challenge; this is simply here for your own edification.