The University of Melbourne
School of Computing and Information Systems
COMP90054 AI Planning for Autonomy

Project 2, 2020

Deadline:
Initial submission: Friday 1st May 18:00
Final submission: Friday 8th May 18:00

This project counts towards 7% of the marks for this subject.
This project must be done individually.

Aims

The aim of this project is to improve your understanding of PDDL planning.

https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html

The domain

This assignment relates to pacman domain described at https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html. Briefly, the domain consists of a single player agent *Pacman* that begins at a designated location on a map. The map consists of a two dimensional grid of cells. For this assignment we will assume there are no walls restricting motion. Pacman can therefore move to any adjacent location and will eat any food or capsules present at that location. The goal is to eat all of the food in the map.

The domain may also include a number of *ghosts*. For this assignment, we assume the ghosts are immobile and will always remain in their initial location. Pacman must avoid moving to a location occupied by a ghost as colliding with a ghost will cause Pacman to die.

The map may also contain one or more *capsules*. After eating a *capsule*, pacman becomes invulnerable for a number of moves. During this time, Pacman may kill ghosts by colliding with them. **This rule is only relevant for the second and third tasks.**
The domain includes only one action - `move`. The first two parameters of this action are Pacman's current location and Pacman's new location after executing the move. In your

implementation, you may also choose to include other parameters after these two as part of your `move` action.

Your tasks

You will need to solve planning tasks in this domain by modelling them in PDDL. Recall that a PDDL model consists of two files: a problem file describing the initial and goal states and a domain file that describes the dynamics of the problem. For each task, you will need to produce a domain file and a problem file. Each task provides a problem map which characterises the initial state you will need to describe in PDDL as part of your problem file. The legend for the problem maps is as follows:

- **P** - Pacman
- **G** - Ghost
- **F** - Food
- **C** - Capsule
- **-** - Empty cell

Each cell in the map is labelled using a row letter (a-b) and column number (1-5). For example, in the Part 1 problem map there is a ghost in location a2.

You may test your implementation using `http://editor.planning.domains`. This site also contains a number of sample PDDL models for different problems, which you may wish to review to familiarise yourself with the syntax.

Create a private repository (0 marks)

Please use the gitlab account you created for assignment 1 to fork the following repository into your own git repository as this contains the framework code you need to modify: `https://gitlab.eng.unimelb.edu.au/cewin/comp90054-2020-a2`. **Make sure you select a 'Private' repository**. This will ensure your repository can only be accessed by you and by the teaching staff, otherwise your repository may be viewable by others in the class.

Once done, please fill in this form as we will use the details you submit to find your project for marking.

These tasks must be completed by the initial submission date.

Part 1 (3 marks)

Problem map

```
  | 1 | 2 | 3 |
 -|---|---|---|
 a| P | G | F |
 b| _ | _ | _ |
 -|---|---|---|
```

Task

Assuming that capsules do not exist, create the domain and problem files that allow a PDDL planner to produce a plan that causes pacman to eat all of the food in the map, while avoiding the ghosts. The file names of your domain and problem files must be `part1_domain.pddl` and `part1_problem.pddl` respectively.

Expected Solution

(move a1 b1) (move b1 b2) (move b2 b3) (move b3 a3)

`Note:` Your move actions may include other parameters after the first two

Part 2 (2 marks)

Problem map

```
  | 1 | 2 | 3 | 4 | 5 |
 -|---|--- ---|---|---|
 a| P | _ | _ | G | F |
 b| C | _ | _ | G | _ |
  |---|---|---|---|---|
```

Task

For this task we introduce capsules. Assume that the effects of the capsule last forever, so Pacman can eat ghosts at any time after consuming the capsule. Create the domain and problem files that allow a PDDL planner to produce a plan that causes pacman to eat all the food in the map (as in part 1). For this task, there is no requirement that Pacman eats all of the ghosts on the map, however in this problem it will need to eat at least one to get to the food. The file names of your domain and problem files must be `part2_domain.pddl` and `part2_problem.pddl` respectively.

Expected Solution

(move a1 b1) (move b1 b2) (move b2 b3) (move b3 b4) (move b4 a4) (move a4 a5)

`Note:` Your move actions may include other parameters after the first two

Part 3 (2 marks) (More Challenging)

Problem map

```
  | 1 | 2 | 3 | 4 | 5 |
 -|---|--- ---|---|---|
 a| P | _ | _ | G | F |
 b| _ | C | _ | G | C |
  |---|---|---|---|---|
```

Task

For this task, pacman can kill the ghosts after eating the Capsule **but** the effects of the Capsule will wear off two moves after it is eaten. Additionally, pacman **must** now eat **all** of the ghosts before eating the last piece of food on the map. Create the domain and problem files that allow a PDDL planner to produce a plan that causes pacman to satisfy these conditions. The file names of your domain and problem files must be `part3_domain.pddl` and `part3_problem.pddl` respectively.

Expected Solution

(move a1 b1) (move b1 b2) (move b2 b3) (move b3 b4) (move b4 b5) (move b5 b4) (move b4 a4) (move a4 a5)

`Note:` Your move actions may include other parameters after the first two

Checking your submission

You should use `http://editor.planning.domains` to verify that your PDDL model produces a valid plan.

Marking criteria

This assignment is worth 7% of your overall grade for this subject. Marks are allocated according to the breakdown listed above. Tutors will review each submission to ensure correctness, so passing the provided test cases does not guarantee full marks. No marks will be given for code formatting, etc.

Originality multiplier

We will be using a code similarity comparison tool to ensure that each student's work is their own. For code that is similar to another submission or code found online, an originality multiplier will be applied to the work. For example, if 20% of the assessment is deemed to have been taken from another source, the final mark will be multiplied by 0.8.

Late submission policy

Submissions that are late will be penalised 1 mark per day, up to a maximum of 5 marks."

Submission

Initial Submission (0 marks)

By the initial submission date, you must have completed the tasks in the 'Creating a private repository' section. On this date we will clone your repository to ensure that we are able to access it.

Final Submission

The master branch on your repository will be cloned at the due date and time.

From this repository, we will copy *only* the files: `part1_problem.pddl`, `part1_domain.pddl`, `part2_problem.pddl`, `part2_domain.pddl`, `part3_problem.pddl` and `part3_domain.pddl`. Do not change any other file as part of your solution, or it will not run. Breaking these instructions breaks our marking scripts, delays marks being returned, and more importantly, gives us a headache.

Note: Submissions that fail to follow the above will be penalised.

Academic Misconduct

The University misconduct policy[1] applies. Students are encouraged to discuss the assignment topics, but all submitted work must represent the individual's understanding of the topic. The subject staff take academic misconduct seriously. In the past, we have prosecuted several students that have breached the university policy. Often this results in receiving 0 marks for the assessment, and in some cases, has resulted in failure of the subject.

Important: As part of marking, we run all submissions via a code similarity comparison tool. These tools are quite sophisticated and are not easily fooled by attempts to make code look different. In short, if you copy code from classmates or from online sources, you risk facing academic misconduct charges.

But more importantly, the point of this assignment is to have you work through a series of foundational search algorithms. Successfully completing this assignment will make the rest of the subject, including other assessment, much smoother for you. If you cannot work out solutions for this assignment, submitting another person's code will not help in the long run.

[1] See https://academichonesty.unimelb.edu.au/policy.html