Informatics 2D Coursework 2: Planning with PDDL

Mattias Appelgren and Alex Lascarides

Deadline : **3pm, Thursday 25th March 2021**

Introduction

This assignment is about planning using the **Planning Domain Definition Language (PDDL)**. It consists of 3 parts:

- Part 1 - A written exercise in which you will formalize a planning problem using **PDDL** (worth 25%).
- Part 2 - Implementing and verifying the correctness of your written model using a PDDL planner (worth 20%).
- Part 3 - Extending the model to deal with additional complications in the environment (worth 55%)

This assignment is marked out of 100, and it is worth 15% of your overall grade for Inf2D.

The files you need are on the Inf2D Learn website: click on Assessment (on the LHS menu), then "Assignment Submission" and then on "Coursework 2 Symbolic Planning" or you can access it from this url:

`https://www.learn.ed.ac.uk/webapps/blackboard/content/listContentEditable.jsp?content_id=_4780523_1&course_id=_80286_1`

You will download a the file `coursework2.tar.gz` which can be unpacked using the command:

```
tar -xf coursework2.tar.gz
```

This will create a directory called coursework2 which contains a couple of example pddl files, the MetricFF planner, answers.txt where you will put your answers to question 1, and a README file.

Please be sure to read the README file carefully as it involves instructions for setting up and running the **MetricFF planner**, which will be used in parts 2 and 3.

Submission

Create a directory in which you keep the files you submit for this assignment. This directory should be called `Inf2d-ass2-s<matric>` where `<matric>` is your matriculation number (e.g 1234567).

In this directory, write your answers to part 1 and part 4 in a file you call `answers.txt`. For the implementation in parts 2 and 3, copy and edit the corresponding `*-template.pddl` files, available in the coursework archive.

Submit your assignment by compressing your directory into a file `Inf2d-ass2-s<matric>.tar.gz`. You do this using the following command in a DICE machine:

```
tar -cvzf Inf2d-ass2-s<matric>.tar.gz Inf2d-ass2-s<matric>
```

You can check that this file stores the intended data with the following command, which lists all the files one will get when one extracts the original directory (and its files) from this file:

```
tar -t Inf2d-ass2-s<matric>.tar.gz
```

Also, you can check the size of the above file using the following command, to check that the file stores the intended data:

```
ls -l inf2d-ass2-s<matric>.tar.gz
```

Submit this file via LEARN by uploading the file using the interface on the LEARN website for the course Inf2D. If you have trouble please refer to this blogpost:

`https://blogs.ed.ac.uk/ilts/2019/09/27/assignment-hand-ins-for-learn-guidance-for-students/`

The deadline for submission is **3pm, Thursday 25th March 2021**.

You can submit more than once up until the submission deadline. All submissions are timestamped automatically. Identically named files will overwrite earlier submitted versions, so we will mark the latest submission that comes in before the deadline.

If you submit anything before the deadline, you may not resubmit afterward. (This policy allows us to begin marking submissions immediately after the deadline, without having to worry that some may need to be re-marked).

If you do not submit anything before the deadline, you may submit *exactly once* after the deadline, and a late penalty will be applied to this submission, unless you have received an approved extension. Please be aware that late submissions may receive lower priority for marking, and marks may not be returned within the same timeframe as for on-time submissions.

For additional information about late penalties and extension requests, see the School web page below. Do **not** email any course staff directly about extension requests; you must follow the instructions on the web page.

`http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/`
`late-coursework-extension-requests`

Good Scholarly Practice: Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

`http://www.ed.ac.uk/schools-departments/academic-services/`
`students/undergraduate/discipline/academic-misconduct`

and at:

`http://web.inf.ed.ac.uk/infweb/admin/policies/`
`academic-misconduct`

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

Part 1: Modelling The Domain (25%)

Problem Description

King Minos has been demanding sacrifices to feed the Minobot - half bull half robot. This year BoTheseus, hero of Athens, has volunteered to slay this mighty foe. However, to succeed in this heroic deed, BoTheseus must be programmed to slay the beast. In this coursework you will model Minobot's Labyrinth and BoTheseus' actions in the Planning Domain Definition Language (PDDL). You will implement the planning domain and execute it using a PDDL planner. To be successful in his quest BoTheseus must have the capability to move through the Labyrinth (pictured in Figure 1). He must also be able to carry his sword to the Minobot's lair where he will slay the mighty Robo-bull.

Task 1.1: Describing The World State (5%)

Your first task is to create the predicates which will describe an atomic state in your planning domain. Your model should be able to describe the following:

1. Minobot's Labyrinth, i.e. the layout of the Labyrinth and therefore which rooms are *connected*
2. The location of BoTheseus, Minobot, and objects such as BoTheseus' sword

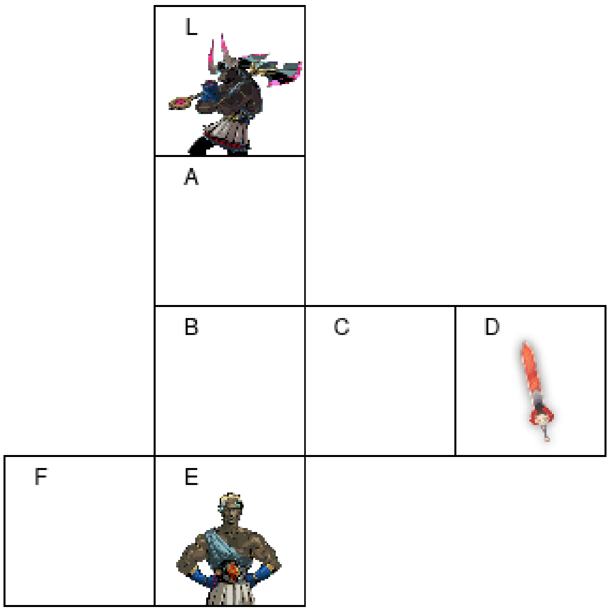

Figure 1: Minobot's Labyrinth

3. Whether or not the Minobot has been *defeated*
4. When BoTheseus is *holding* an object

Name all predicates which you will use to describe the world as well as their purpose. If you need additional predicates to model the actions in task 1.2 please also describe them here as well.

Once you have defined your predicates write down an initial state where BoTheseus begins on the **E**ntrance, the Minobot is in the **L**air, and the sword is in chamber **D**. The Minobot should not be defeated. Additionally write down a goal where the Minobot has been defeated and BoTheseus ends at the **E**ntrance.

Ensure the initial state is a valid PDDL initial state which could be used to create a plan for completing the problem given the actions in Task 1.2.

Task 1.2: Actions (10%)

In this question you will define the actions BoTheseus can perform. Define the arguments, preconditions and effects of each of the actions.

BoTheseus can *Move* between two locations if they are connected. You cannot move diagonally.

BoTheseus can *pick up* objects, such as the Sword, if it is in the same location as the object and isn't holding anything else. That is, it can only hold one object at any one time.

Finally, BoTheseus can *Slay* Minobot if BoTheseus is holding his Sword and is in the same location as Minobot.

Task 1.3: Backwards state space search (10%)

In this question you must perform the backwards state space search algorithm as described in the lectures.

The starting state is one where BoTheseus is in chamber **B**, the Sword is on the ground in location **A**, and the Minobot is in the **L**air. The goal is for the Minobot to be defeated.

Spell out the working of the algorithm by specifying at each step which are the *relevant* actions and what the current *goal state* is. State why the relevant actions are selected as relevant.

State clearly which action was selected at each step. For the purpose of this task you may assume the search always picks the correct action when there is a choice, so you do not need to simulate dead ends.

Finally, state clearly why the algorithm terminates when it does and state the plan which was selected.

Part 2: Implementation (20%)

Implement your domain model in PDDL using the example files provided in the course work hand out as a starting point. Make sure to read the README file included with the coursework which provides instructions for setting up and running the FF planner.

Test your model by creating problem files for the following problems:

Task 2.1 (15%)

1. BoTheseus begins in **E**ntrance
2. The sword is in chamber **D**
3. Minobot is defeated

Task 2.2 (5%)

1. BoTheseus begins in **E**ntrance
2. The sword is in chamber **D**
3. Minobot is defeated
4. BotTheseus ends at the **E**ntrance

Part 3: Extension (55%)

In this part you will extend the domain to support additional actions and considerations.

Task 3.1 Minos Strikes Back (15%)

King Minos has heard of our plans of defeating the Minobot and has asked Deadalus to add additional defences to the Labyrinth. In particular he has added a locked gate between chamber A and the Lair and he has upgraded the Minobot's protective leather plating such that it cannot be defeated without the sharpest of swords, and BoTheseus' sword is blunt, and therefore would be unable to cut through Minibot's protective hide.

Autoadne, who has fallen in love with BoTheseus has managed to smuggle a key into the Labyrinth. BoTheseus can open the gate if he is in a chamber adjacent to it (e.g. A or Lair) and if he's holding the key. Remember that BoTheseus can only hold one item at the same time.

Autoadne also managed to smuggle in a whetstone which can be used to sharpen the sword. BoTheseus can sharpen the sword if he is holding the sword and is in the same chamber as the whetstone. He can now only slay Minobot if the sword is sharpened.

You will need new actions and predicates to deal with these changes and you will have to change some of the old actions. Create a new domain file with these new additions.

Then create a problem file for your new domain such that BoTheseus begins on the **E**ntrance, the (blunt) sword is at in Chamber **D**, the key is in **F**, the whetstone is in the **E**ntrance, Minobot is in the **L**air and there is a locked gate between **A** and **L**. The goal is for the agent to have defeated Minobot.

Task 3.3 Yarn (20%)

BoTheseus has realised that keeping track of where he is in the labyrinth is a constant struggle and he worries that he will get lost. Luckily Autoadne has an excellent idea, namely to use yarn to keep track of where BoTheseus has walked so that he can find his way back to the start.

Autoadne has left a couple of balls of yarn in the Labyrinth. From now on BoTheseus will only walk into a location if it is marked with yarn or if he his holding a ball of yarn (which has enough yarn to mark the new location). Walking into an unmarked location while holding a ball of yarn will mark the new location but there is only enough yarn in each ball to mark 3 locations. BoTheseus can move freely into locations which are marked with yarn. Holding a ball of yarn works the same as holding other objects (i.e. only one at a time).

Test these changes on a problem where

1. BoTheseus begins on the **E**ntrance
2. the blunt sword starts on **D**
3. the whetstone is in **E**
4. the key is in **F**
5. the Minobot on the **L**air
6. there's a locked door between **A** and **L**
7. there's a ball of yarn in **E** and one in **C**

The goal is to defeat Minobot and have BoTheseus end on the **E**ntrance.

Task 3.4 Autoadne Joins the fight (20%)

Navigating the labyrinth without the markings left behind by yarn is difficult, and BoTheseus realises that he doesn't actually know where he's going without the markings on the ground. It would require someone who actually knows where they are going to first lay out the yarn for BoTheseus to follow. Luckily Autoadne knows what the labyrinth looks like so she can help BoTheseus.

In this problem Autoadne becomes an agent who's actions you can control. Autoadne and BoTheseus have the same capabilities as each other with the only difference that only BoTheseus can slay the Minobot and only Autoadne can mark the ground with yarn (in other words, only Autoadne can move into a location without yarn. She can do this while holding a ball of yarn (which has not run out) and doing so will mark the location she walks into).

Add Autoadne as a second agent to the domain and update the actions so that both agents can be made to perform actions. Test your domain on a problem where:

- BoTheseus and Autoadne start in **E**
- the blunt sword is in **D**
- there are two balls of yarn, one in **E** and one in **A**
- a key is in **F**
- there is a gate between **A** and **L**
- the whetstone is in **B**
- the Minobot is in **L**

The goal is for the Minobot to be defeated.