Artificial Intelligence

Planning Systems Coursework
Assigned: 6 October
Due: Tuesday, 13 October, 7h30pm

Prof. Felipe Meneguzzi
João Paulo Aires (assistant)
Maurício Magnaguagno (assistant)

October 8, 2015

1 Planks Domain

You must work in this project **individually**. You are free to discuss high-level design issues with the people in your class, but every aspect of your actual formalisation must be entirely your own work. Furthermore, there can be no textual similarities in the reports generated by each group. Plagiarism, no matter the degree, will result in forfeiture of the entire grade of this assignment.

For this assignment you will be using the open source implementation of Graphplan available with the JavaGP planner[1]. You can obtain a copy of this software from its SourceForge project site, or you can download a more conveniently packaged version of the software from our Moodle area. Your deliverables must be handed in via Moodle using the correspondingly named upload rooms. At the end of this assignment, you will upload **one zip file** containing the problem specification files in the PDDL format, specifically:

- one file named `planks.pddl` containing the domain encoding;
- five files containing the problems you formalised, named `pb`σ, where σ is the sequential number of your problem, e.g. `pb1.pddl`, `pb2.pddl`;
- the same number of files (with similar names) showing the traces created by JavaGP for the problems you created (e.g. `pb1.pddl` and `pb1.txt`);
- any additional problems you created; and
- one report following the guidelines detailed below.

If you make the bonus part, your submission must also contain:

- (bonus) a separate file named `planksSuper.pddl` containing the bonus domain; and
- (bonus) the same problems, formalised for the bonus domain.

2 Overview

In this assignment you will formalise the Planks domain, which consists of multiple agents isolated in an archipelago of small islands that are divided by a fast flowing river. Agents in this game can move around whenever there are *planks* connecting islands, basically makeshift bridges made out of these planks strewn across the islands, so they can hop between islands in order to reach a predefined destination island. Thus, there are three types of object to be modelled in your domains: "agents", "islands" and "planks". Agents can only "lay" planks to connect "adjacent" islands (which they need

[1] `http://sourceforge.net/projects/emplan/`

to cross from one island to the next), and can only "pick up" planks that are themselves connecting the island currently occupied by themselves. Since planks are heavy, but not very sturdy, agents cannot cross bridges while carrying a plank (otherwise they will break the plank they are trying to traverse, drowning in the process). The islands themselves are too tiny to contain more than a single agent, and since agents cannot swim, they can never move into an island already "occupied" by another agent.

Below, we illustrate the skeleton of a PDDL domain file (you can find additional domains in Moodle):

```
(define (domain planks)
  ; You may need to add more requirements
  (:requirements :strips)

  (:predicates
    ; TODO predicate definitions
    ; Example (at ?agent ?place)
  )

  (:action cross
    :parameters (?par1 ?par2 ?par3 ?par4)
    ; TODO precondition and effect
  )

  (:action pick
    :parameters (?par1 ?par2 ?par3 ?par4)
    ; TODO precondition and effect
  )

  (:action lay
    :parameters (?par1 ?par2 ?par3 ?par4)
    ; TODO precondition and effect
  )
)
```

As a **bonus** to this assignment, you must model super planks variation of the domain, whereby bridges are extra heavy and require two agents to move them between islands. Here, the "pick up" action now requires two agents (on either sides of each plank) to lift a plank and move it towards an adjacent set of islands.

Whenever we need to use automated tools to solve problems on our behalf, we must provide a consistent specification of the *transition system* of the underlying problem. If we specify the problem poorly, we jeopardise the planner software ability to generate valid responses, leading to false negatives and unnecessarily long waiting times for the planner at best, or incorrect plans at worse. Thus, specifying the PLANKS domain in PDDL gives you a chance to develop your skills in designing consistent transition systems, helping you avoid bugs in the software you will write in the future to handle all kinds of other processes.

Your assignment is to develop a domain file from the specification above, and then model the situations depicted in the images below as individual problem files. The following hints may be useful, but you are welcome to use your creativity as long as you adhere to the specification mentioned above:

- You need actions for moving between islands;
- You need actions to pick up and lay planks between islands, so agents can take move around (There is no sensing involved here; if a plank connects the island currently occupied by the agent to any other island, then he can pick up that plank.);
- Remember that when modeling pick up and lay actions, you may need an "unloaded" predicate to indicate you are carrying nothing (otherwise your planner would need a quantified negation, which JAVAGP does not have);
- (If you do the bonus task) You need an action representing the joint pick up and lay of heavy planks; and

- Look at the words in quotation-marks in the specification above. They may give you a good skeleton to base your domain on.

3 Problem Instances

Below are images of the problem instances that you need to model in PDDL, once you are done making your domain file. The caption that accompanies each image should be fairly self-explanatory; remember, you must model connections between islands for the players to move from one to the other, and you can only move between a islands that are connected and have a plank linking them. In the instances shown in Figures 1 through 4, any locations that share an edge can be considered connected. Problems 1 and 2 should guide you in your first steps, while problems 3 and 4 are the complex problems proposed to be solved by your description. Locations that share only corners and no edges **are not** connected. Note that missing elements in the goal image can be anywhere, declare only the agents and planks you see in the goal.

Figure 1: Problem 1 - Cross the bridge.

Figure 2: Problem 2 - Pick and lay the plank.

Figure 3: Problem 3 - Reach the island at the right.

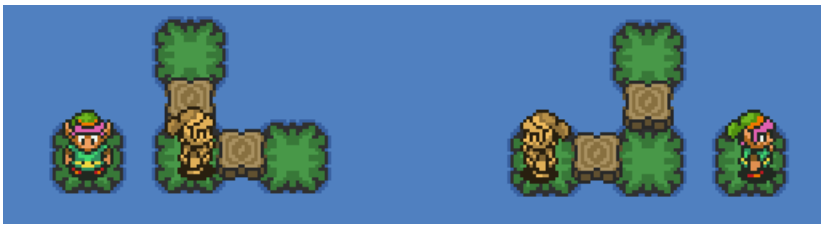

Figure 4: Problem 4 - Move left agent to the right island, and the middle agent to the left island.

Notice that the above pictures give you an idea of the initial state of the world (which you must encode in your PDDL problem file). They also tell you what the goals are - Link's[2] final location (in green), and the various planks that must be placed on the way. If you look at the syntax of example PDDL problem files, you will see that these are the three main parts of a problem file - (1) the objects, (2) the initial state, and (3) the goals.

[2]We really know only one character, sorry for the lack of creativity. We welcome sprite artists for next semester's challenge, full credits will be given.

4 Grading

In order to properly evaluate your work and thought process, you will write a 2-page report in the AAAI conference format explaining your encoding and experiments. These guidelines are to be followed **exactly**. **Reports that are less than two pages of actual content, or not in format will receive 0 marks for the report criterion.** This report will be included in the deliverables of the assignment. The formatting instructions are available at the AAAI 2015 website[3]. The report must have the following sections:

- An introduction with your understanding of the problem domain, outlining the remainder of the paper;
- Two domain formalisation sections explaining your approach to formalising the problems from Section 3
- One experimentation section where the performance of JavaGP is measured using your action formalisation for each of the domains, on multiple problems.
- One conclusion section, where you will summarise your experience in encoding planning domains and discuss the performance of JavaGP, and any limitations encountered in solving the problems you encoded.

Grading will take consider elements of your encoding, experimentation and reporting of the work done. The criteria, as well as their weight in the final grade is as follows:

- Domain Encoding (30%) — correctness of the domain encoding, in relation to the domain specification from Section 2;
- Problem specifications (basic) (20%) — correctness of the problem specifications for problems 1 and 2 used for the experiments, particularly the initial state specification, as missing predicates here will jeopardise JavaGP's ability to solve your problem;
- Problem specifications (full) (10%) — correctness of the problem specifications for problems 3, 4 and the secret problems used by the instructors;
- Overall report readability (20%) — how accessible and coherent your explanation of your encoding is;
- Experiments (20%) — how coherent the proposed experiments are in measuring the performance of JavaGP, notice that this criteria is complementary to the one regarding problem specification.
- Heavy planks (20% bonus) — if you have finished a working version of the heavy planks domain in which two agents are required to move a single heavy plank.

5 Sample PDDL files

Here are some sample files for other domains that you should look at before attempting to encode your own domain file and problem files. These will give you an idea of the information that goes into each of those files, and the syntax. You can also use these samples to test out the running of the planner, and to see what the output should look like.

- Blocks World Domain
- Gripper domain

[3] `http://www.aaai.org/Conferences/AAAI/2015/aaai15call.php` (the template is in the "Author Kit" link)

6 Miscellaneous Advice

Here are some lessons we learned in creating our own solution and writing papers/reports:

- JavaGP does not parse "or" conditions or effects. You need to specify conditions / effects as a list of predicates bound together by a single "and".
- As a first step, you should at least look at the sample domain and problem files given at the beginning of this specification. Additionally, here is a link to PDDL domain and problem files (for scenarios different from the one you need to model in this project), from past iterations of the International Planning Competition (IPC).
- The best way to figure out how to model a domain and associated problems is to look at these examples. If you feel the need for documentation, here is a paper that talks about the complete PDDL specification, with BNF specification at the end (Appendix A): PDDL 2.1 Specification
- Using type predicates increases memory usage and slows down the creation of ground atoms but speeds up the (dominant) time to solve the problem (PDDL, however, allows typing of parameters);
- Tables and graphs are a useful tool to show runtime performance of software[4];
- In order to evaluate the performance of a planning encoding, you need to specify problems with most of the parameters locked in, and measure runtime as one parameter increases (e.g. number of locations, number of containers, etc);
- Pasting your entire domain specification (or problems) into the paper does not count as content (now you cannot say you were not warned);
- Overly large figures used to simply fill space in the report are also not a good idea;
- Reviewers have a more pleasant reading experience when papers are generated using LaTeX, it is very easy to spot the difference.

7 Sample Solutions

The solutions are using objects with the names we established, if you use different object names keep in mind to compare only the action names. The correct plan for the problem in Figure 3 using the instruction's formalization is:

```
cross(link, plank1, px1y2, px2y2)
pick(link, plank1, px2y2, px1y2)
lay(link, plank1, px2y2, px3y2)
cross(link, plank1, px2y2, px3y2)
```

[4] GnuPlot is an excellent (and free) graph making software, available at `http://www.gnuplot.info/`