The University of Melbourne
Department of Computer Science and Software Engineering
COMP90054 Software Agents

Project 2, 2015

Set: 16 September.
Preliminary project presentation: in lecture, 24 September 2015
Preliminary Electronic Submission: 11am Monday 19 October 2015
Final project presentation: Week beginning 19 October 2015
Final Electronic Submission: 4pm Wednesday 21 October 2015

This project counts towards 40% of the marks for this subject (Electronic submission: 30% and a final Presentation: 10%) and can be done either individually or in small groups.

Aim

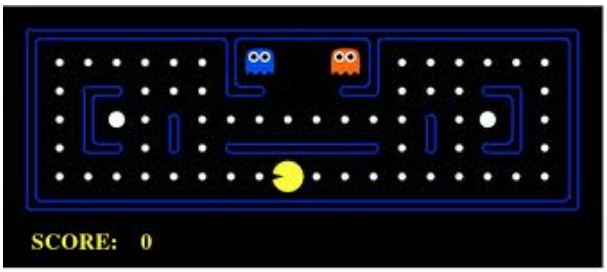
The purpose of this project is to implement a *Pac Man* planner that can play and compete in a *tournament*. To help you develop your solution you must express the Pac Man domain using two different approaches, including

(i) a PDDL formulation of Pac Man based on classical planning principles (5 marks), and

(ii) a situation calculus axiomatisation of Pac Man which facilitates, among other things non-deterministic programming principles (5 marks).

(iii) Finally, you must provide a working Pac Man planner that is capable of playing Pac Man and competing in the tournament. Your planner can use any technique, or combination of techniques, that you choose. For example the *FF* planner, a variant of the *Golog* planner or a purpose built planner of *your own making*. Marks: 30, allocated as follows:

- Working PacMan competitor (10 marks)
- Consistently beats the baseline agent (10 marks)
- Final competition place (10 marks)



The intention of the project is to follow on from your project 1 submission concerning Berkeley Pac Man as a basis and starting point for project 2, based on `http://ai.berkeley.edu/contest.html`.

Your task

1. Axiomatisation of Pac Man in PDDL (classical planning approach)

Typical applications of planning consist on one or several calls to a planner. The instances are generated *on the fly* by a *front–end* (the pacman engine), and the solutions (plans) are interpreted as executable instructions. As the pacman is not a classical single agent problem, you are required to implement two points of view: The point of view of the pacman, where its goal is to stay alive while eating all the dots of the grid, and The point of view of the ghost, whose goal is to kill pacman. Assume that the game is turn-based, so at each step an instance is generated with the current state of the world, i.e. the dots and ghosts locations in the grid. From the point of view of pacman, the ghosts don't move, and vice-versa, that is, the environment is static.

At each step the planner would come out with a plan to eat all the dots while avoiding static ghosts, and plans to enable ghosts to kill the static pacman. A simple interpretation of the plans by the pacman engine is to execute only the first action of the plan, ignore the remaining actions, and call the planner in the next step with a new updated instance accounting for the new locations of the ghosts and the pacman.

The axiomatisation should define the STRIPS model for pacman, and the STRIPS model for a ghost. Explain clearly the assumptions made, e.g. pacman do not move to cell X, when Y holds, etc., and describe several initial states or goals to illustrate interesting situations.

2. Axiomatisation of Pac Man in the Situation Calculus (non-deterministic programming approach)

In this step, you must define Pac Man using the situation calculus, creating the axioms that define the domain—including the *fluents*, *actions* and *successor state axioms*.

3. Implementation of Pac Man using *either* a classical planner *or* non-deterministic program

The STRIPS model can easily be written in PDDL, encoding one domain file for pacman, and one domain file for the ghost containing the predicates and the actions of the world. The problem file describes the initial state and goals. Therefore, with a single domain for either the pacman or the ghost, several problems can be generated by only updating the problem file.

Use the engine of the tournament to generate the PDDL files at each step that an action is required, call your favourite planner, and parse the solution in order to choose the best action.

Different domains can be used to encode different strategies. Please explain each domain encoded and the differences among them, if needed. Note that the pacman tournament has different rules as it's a two teams game, where your Pac-mans become ghosts in certain areas of the grid. Please read carefully the rules of the pacman tournament.

You can alternatively implement a non-deterministic programming approach based on the *IndiGolog* planning language, based on your axiomatisation in the situation calculus.

You are also free to implement a planning competitor in the language of your choice, for this option please check with us first.

Tournament

Please submit your projects early, as we will be running tournaments based on the submissions in the weeks before the project presentations. Note you can re-submit multiple times.

Deliverables

You must submit

(i) a PDDL formulation of Pac Man, using *PDDL* (*.pddl*) file format

(ii) a situation calculus axiomatisation of Pac Man, in *Prolog* (*.pl*) file format

(iii) a working Pac Man *planner* that is capable of competing in the tournament (e.g. either in Python (*.py*) or Prolog (*.pl*) format, or possibly another programming language (*please check with us first*).

(iv) You must include a *readme.txt* file which briefly documents the code and briefly outlines the approach that you took to implement the domain

(v) a *group.txt* plain text file listing your group members' logins, one per line

In your README.TXT, you must describe the techniques that you have implemented in your planner for the tournament. Please clearly state any assumptions that you make. If possible, include an analysis of the strengths and weaknesses of your planning technique.

The `group.txt` file must (strictly) include a list of lowercase login names, one per line for all people in your Project group (or just one login for individuals). For example,

```
>cat group.txt
>login1
>login2
```

Presentations

You or your group will be required to give two oral presentations: one early trial presentation (not marked) presentation where you will have the opportunity to describe and discusses your project; and one final (marked) project presentation to give you an opportunity to describe your results and/or findings.

Submission

One member of the group must submit *all* files electronically, including your domain definition (PDDL and Golog) files and the README.TXT report. Please include all the PDDL files if a classical planner is used, along with the code to execute your team. Please include all the Prolog files, if a Golog variant is used.

In the event of group work, please include your names in the report and source code. Use the following command to submit your work on e.g. *dimefox*

submit COMP90054 2 <teamname.zip>

Important: please follow the following specific steps for submission.

1. Your code must live exclusively within the teams/¡yourteam¿/ subdirectory, which should be a python package (and thus include `__init__.py`).
2. Your code will be run by the following command:
   `python2.7 capture.py -r teams/<team1>/team.py -b teams/<team2>/team.py`,
   please make sure your `AgentFactory` is defined in team.py.
3. Attach a README.txt with your names and emails, so we can send you specific instructions on how to change the code so we can run it, if needed
4. zip your folder, and submit a single zip file of your team directory: `cd teams; zip -r teamname.zip teamname/`.
5. All the teams that use a planner, and generate pddl fiiles and solutions, please redirect your planner call, solution outputs, etc. to your own folder (Do *NOT* use the current working directory to write temporary files). You can use python code to do it automatically, or you can hardcode it assuming that your team will be located in ./teams/yourteamname/ folder.
6. You may assume that `./ff` is version 2.1 of the Metric-FF planner (`https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html`). If you want to use any other planner or 3rd-party executable please discuss with Toby before submission.