Project 2: Functional Programming

Proposal Due: 22 November; Final Project Due: 2 December

You must do this project in a group of size 2 or 3. You cannot do it alone. Make sure all members of the group understand and can explain your solution.

The project is to build a feasibility study to investigate whether functional programming is suitable for some task. You are to have a program that works, and does aspect in-depth, going beyond what has been covered in the course and assignments. This "something extra" should be something you are proud to show off. You also need to draw a conclusion about the feasibility of what you have investigated.

When deciding whether to adopt a new technology for a task, it is important to do a feasibility study, to critically evaluate whether the technology is up to the task and whether it promises to be better than the traditional technologies or not. Your job is to investigate whether logic programming is suitable for some task that you choose.

The project is not meant to be complicated. It should be about the same work as two assignments.

There are a few deadlines:

- You are to have teams and a proposal by November 22. Your proposal should be discussed with a TA before then, to make sure that what you are suggesting is feasible and not trivial (given what we have covered). The project proposal should be on the ubc wiki (create a link from `http://wiki.ubc.ca/Course:CPSC312-2016`), where other students can see your proposal[1]. You are encouraged to coordinate with other groups, and even share code, but each group is responsible for their own in-depth aspect. You are welcome to use Piazza to build teams. (2 or 3 in teams means that any team can absorb new members; a team of 3 can split into two teams. Teams can work together.)

- A final project is due on December 2nd. Each project will be highlighted on the UBC wiki, so that everyone can see everyone else's project. This write up should include what you have done and your conclusions based on the evidence you have. You will be expected to post your code on handin by the deadline.

[1] If you would prefer to not have your description on the wiki, submit a description to handin, and notify the instructor. There is no penalty for not participating in the wiki.

- During the first two weeks of December, you will need to make an appointment with a TA to demonstrate your project. You should expect the TA to run the program (with your team present), while you explain what is interesting about what you have done. They will use some test cases that you provide and some of their own. You should also expect that the TA will inspect your code to see whether you have good coding practices (including giving the intended interpretation for all symbols), and look at your conclusion. The TA will determine your grade based on this demonstration. All projects need to be marked by December 14. There is an extended period to do demos because we want to be flexible and fit in with people's schedule, not so you can have more time to work on it.

Submit your Haskell code in text files to handin using the directory `cs312/project2`. This Haskell program should work with ghci. It should link to the description on UBC Wiki.

Assessment

Your assessment will be similar to Project 1. A detailed grading rubric will be posted on the Wiki.

Suggested Topics

If in doubt talk to a TA or the instructor. We would rather help you earlier to have a great project than to evaluate a project that is not as good as it could have been. You should do a web search to see what currently exists, and build on that if it helps. (You must explicitly reference everything that you build on or code that you use. You need to respect copyright.) Combining existing tools cannot count as the in-depth part of the project.

Games

In class we will be covering games.

We plan cover 3 games in detail:

- Magic-sum game (tic-tac-toe)
- Blackjack

- Some grid games where an agent moves and collects rewards and punishments, including a treasure-hunting game where the player needs to explore, collect rewards, avoid monsters etc.

We will cover 2 main strategies:

- Minimax and search for fully-observable games
- Reinforcement learning

Some of the tings you could do:

- Represent other, more interesting games, and test the algorithms on these.
- Build a GUI for some games
- Cover more interesting strategies, e.g., alpha-beta pruning, limited search with an evaluation function, creating features for feature-based reinforcement learning.

Other suggestions

- Some suggestions will be posted on Piazza.
- Reimplement your prolog project in Haskell. It would be interesting to see what can be done easily in each language.
- See Appendix F of the Textbook. There are a number of project suggestions there.

Your suggestion

Suggest another project about something that you are passionate about, you have done in another language or you just want to investigate. Be creative! In the proposal stage, tell us what exists already.