CPSC 312 Functional and Logic Programming Spring 2024

Project 1: Functional Programming

Proposal Due: 12 February; Final Project Due: 26 February

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 one 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 functional 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 February 12. 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 `https://wiki.ubc.ca/Course:CPSC312-2024`), 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 the Canvas discussion list 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 February 26 (yes, this is the same day as the second midterm, but doing the project is the best way to study for the midterm.) Each project will be highlighted on the UBC wiki, so that

[1] If you would prefer to not have your description on the wiki, please email the instructor and TAs. There is no penalty for not participating in the wiki.

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 a link to your code on the wiki by the deadline.

- During the week of February 27 - March 4, you will need to make an appointment with a TA or TAs to demonstrate your project. You should expect to demo the program (with your team present), while you explain what is interesting about what you have done. You will use some test cases that you provide and some of that the TA provides. You should also expect that the TAs will inspect your code to see whether you have good coding practices (including giving clear documentation for all functions and constructors), and look at your conclusions on the wiki. The TAs will determine your grade based on this demonstration.

  Everyone in your group should be able to explain how the project works. You should expect the TA to ask what each member's contribution to the project was; the TA may give different grades for each person in the group if the work was not equitable.

Submit your Haskell code in text files that can be retrieved from the wiki (e.g., on github). This Haskell program should work with ghci.

Assessment

Your assessment will be based on the code during the demo and the writeup on the feasibility of your approach. The TA will assess:

- Does the code work? They will first test it on test cases you suggest, and then test it on their own test cases.
- Is the extra part interesting? Does it give evidence for the usefulness of what you have done? The extra part need not be complicated; indeed simplicity will be rewarded.
- Is the code readable and well documented? Does it give the intended meaning for all functions and constructors? Would someone else be able to take your code and build on it?
- Is the conclusion on the feasibility of your approach justified by the evidence?

A detailed grading rubric is on the wiki.

You may use other's code in your program as long as you respect the copyright and are explicit about what code is the work of others, and what is new code. You must attribute the code you use, even if it is not required by the copyright.

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.) Just combining existing tools cannot count as the something-extra part of the project.

Games

Haskell is a good language for games. In class we will give some Haskell code to interactively play games and using AI techniques for solving games.

Machine Learning

Haskell is a good language for specifying machine learning (after all, the aim of supervised learning is to learn a function that makes predictions on data). You could, for example, investigate various algorithms (such as decision-trees or neural networks), explain the learned hypotheses (e.g., add to the learning algorithm an explanation faculty to explain the prediction or how it was reached), expand simple algorithms to more interesting data sets (e.g., continuous attributes), learn how to play games (e.g., in reinforcement learning).

Other suggestions

- Design a crossword helper that given a list of all English words and a pattern such as "?as??ll" returns the list all works that match the pattern. The challenge here is to design efficient data structures so that, given a pattern, the list can be generated quickly.

- A few years ago Kurt Eiselt designed a project, which everyone had to do. You can do this project (or a variant of it). See `https://www.cs.ubc.ca/~poole/cs312/2024/projects/crusher.zip`
- There may project suggestions from other courses; Google is often very useful.
- Implement some algorithms from another course in Haskell (as long as the original isn't in Haskell; you can't count the same material for multiple courses, but you can build on what you did in other courses.
- Projects from previous term are on the UBC wiki. See `https://wiki.ubc.ca/Course:CPSC312-2023`, `https://wiki.ubc.ca/Course:CPSC312-2021`, `https://wiki.ubc.ca/Course:CPSC312-2019`, `https://wiki.ubc.ca/Course:CPSC312-2018` or `https://wiki.ubc.ca/Course:CPSC312-2017`. You can build on one of these, as long as you obey all copyright, and are explicit about what is your work and what is others' work.

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 and don't assume that that the TA will be familiar with what you are talking about.