CPSC 312 Functional and Logic Programming Jan-Apr 2024

Project 2: Logic Programming

Proposal Due: 25 March; Final Project Due: 8 April

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 logic 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 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 March 25. 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 April 8. 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

[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.

based on the evidence you have. You will be expected to post a link to your code on the wiki by the deadline.

- During April 9-12, you will need to demonstrate your project (the sign-up link will be on Piazza). You should expect the TA(s) 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(s) 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 TA 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 Prolog code in text files that can be retrieved from the wiki (e.g., on github). This program should work with SWI Prolog.

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 relations and functions? 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.

Project 1

You can do your Project 1, but in Prolog. This will let you comare two different programming paradigms.

Natural Language interface to some information

There is lots of information on the web. Prolog has good interfaces to much of this information (particularly in the semantic web library). Prolog probably cannot compete in answering questions about everything, but search engines such as Google can't answer questions in narrow vertical domains (where accuracy and completeness are important). Choose some narrow domain, and write a Prolog program to answer questions about that domain. Choose a domain where the information is available.

You just need a feasibility study that does one thing in a non-trivial way. This can be, for example, in the natural language or in reasoning to acquire the information. There are many useful natural language tools available, e.g., Wordnet (`https://wordnet.princeton.edu/`) provides Prolog definitions of English words.

Representing the rules for some regulatory framework

Many regulatory frameworks are rule based. For example, there are many rules about the degree requirements at UBC. Prolog seems like a good tool for this, where we want to have an explicit representation of the rules.

For example, consider a system that takes in a student's transcript, and can give advice on what courses to take to finish a degree. Or a system

that, given some courses a student would want to take, outputs a plan on how they can take the prerequisites in appropriate terms, perhaps including backups for courses that become full. Or a system that knows about degree requirements, a student's transcript and course timetables and demand, and can allocate courses to students when they need some courses to graduate (and may not realize that some other set of courses may also work).

Web Services

For many tasks people want to use the web for, multiple steps are needed to carry out a task. For example, in booking a vacation, a person may need flights, hotels, car rental, tickets to activities, etc., all for days that coordinate. If someone cannot get the best of all of these they need to make compromises, e.g., one choice may have better hotel, but another may have tickets to a preferable event. It is difficult to coordinate all of these activities when you need to go to one site for flights and another for hotels and another for tickets, etc. Write a program that can input a user's goals and preferences, can access web sites to get availability, and suggest preferred packages.

Mathematics tools or tutorial system

In Assignment 5, you implemented a simple symbolic algebra system that manipulated algebraic expressions. It is not a big step to do differentiation, simplification and even integration (although symbolic integration is much more difficult than differentiation). Consider how this could be extended to a more useful tool. This could either be a tool for doing symbolic mathematics autonomously (like Maple or Mathematica), using automatic differentiation for optimizing a model, or teaching mathematics (or something in between, e.g., where the user selects the steps in integration and the computer carries it out).

Games

There are many games that are logic puzzles or strategy games that involve inference. Clue is a classic example of game that is a logic puzzle. Prolog might be suitable for advising what to do.

Learner

Much of machine learning consists of gradient descent in parameter space. Now you know how to do symbolic differentiation, you might be able to write a learner at a higher-level, where you specify the form of the model and Prolog adjusts the parameters automatically. It is not recommended to do this unless you have taken CPSC 340.

Logic Programming System

Prolog may not be the best tool to first learn about logic programming, as many of the design decisions are driven by large applications. You could write a new logic programming language that, for example includes declarative debugging, so that the user does not need to know how an answer was or wasn't produced. It could be more integrated with web site that provide informations.

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.