MASSACHVSETTS INSTITVTE OF TECHNOLOGY

6.945/6.905

Adventures in Advanced Symbolic Programming

Request for Proposals—Spring 2022

Overview

As advertised in the Red Tape Memo, you will do a significant term project for 6.945/6.905. You should form teams of two or three to work on a project. The project will be due near the end of the term.

In your project you will construct a substantial piece of software that illustrates some of the ideas we have examined in this subject.

Your system should address an interesting class of related problems, by building a library of pin-compatible mix-and-match pieces that can be assembled in a variety of ways to cover your intended domain. The main consideration is that incremental changes to the specification of your problem should require only small changes to the construction of a solution.

One way to build in flexibility is to organize your project so that the major subsystems are made of a family of interchangeable parts. For example, each *major* subsystem of your system should admit more than one implementation. You should be able to demonstrate that your system works when constructed by arbitrarily choosing an implementation for each part from a redundant library.

As part of the project you will present a document describing the domain that your system addresses, the primitive pieces, the means of combination, and the means of abstraction by which you construct solutions in your domain. Every good engineer should be able to teach other engineers how to attack similar problems, or to maintain your software, so you will produce clear, readable code and associated documentation that *you could use* to instruct *your future students* in the tools and techniques that you employed.

This Assignment

By the end of next week you must have assembled your team, decide on goals, and hand in a project proposal that allows us to consider the value of your project as a learning experience and the feasibility of your implementation plan. We will try to provide you with useful feedback within a week. Your proposal should include

- A description of the approximate domain of your proposal.
- A plausible decomposition of your system into major components.
- A plausible plan for implementation and documentation, with assignments of parts to team members.

You should make a rough stab at a proposal first, and then a more complete one. Please submit your draft proposal in lecture on Monday, March 28, and a more complete proposal in by Friday, April 1.

Considerations

Your system should be Free Software[1] (free as in speech, not as in beer), so that it can be shared with, used by, and extended by other students in the future. It should run in MIT/GNU Scheme. It may depend on the Scmutils software, if needed. You may incorporate and build on any software that we have examined in class or on problem sets, or any other Scheme software that you can legally obtain and use. For example, you may use any Scheme code that is in the public domain or is labeled as Free Software.

Suggestions for Projects

We present the following list just to stimulate ideas, but you may choose to build anything that both you and we find interesting, and that we believe provides a good learning experience.

- A special-purpose compiler for a domain-specific language that makes it easy to describe a limited class of interesting applications. The compiler should produce code for some specified hardware or virtual-machine targets.
- An extension of the type-inference program that we used to illustrate the use of unification matching. Perhaps your extension can be used to extract useful type information for programs with union types, parametric types, and (ugh!) side effects. (See Exercises 4.14, 4.15, 4.16, and 4.17.)
- A suite of analysis and synthesis tools for a limited domain of engineering applications. This tool should incorporate dependencies and be able to present them on demand, showing how any part of the result was derived from the parts of the user input and the rules/programs used in constructing this result.
- A tool that could be used to help teach some technical subject, such as mechanism design. Such a tool should do more than illustrate an idea: it should be able to answer reasonable questions about the illustrations that it provides, with appropriate justifications for its answers, if pressed.
- A "semantic search" for programs, electrical circuits, or (much harder) for some restricted natural language text, that can allow one to find all of the places in the source that need to be examined or modified, based on matching patterns that respect semantically invariant transformations.

[1] See http://www.fsf.org/licensing/essays/free-sw.html to find out what is meant by "Free Software."