CS 4110 Project Alpha Report

Clay Goodling (jcg284)

December 2020

1 Vision

My current vision for the project is the same as it was when I started. I hope to build a system capable of creating an interpreter for an arbitrary language from a description of its semantics. After this second sprint, I believe that I am on track to have a fully functional system without any bells or whistles. I might be able to clean up a few of the idiosyncrasies (like how to put an equality as a condition for a rule of a relation, you have to write `true = eq a1 a2`), but I don't expect a professional level of polish.

2 Summary of Progress

In this sprint, I finished the script which translates a set of semantics to an Ocaml module and created semantic files for call-by-name and call-by-value lambda calculus. I also reformed the syntax of the semantic metalanguage somewhat to make it easier to implement relations, and I expanded the set of built in relations to all of the basic operations on ints, floats, bools, strings and maps. I then tested the fully generated arith module by manually constructing some basic expressions in utop and stepping them.

3 Productivity Analysis

This sprint felt less productive to me than the last one, but I seem to have made at least as much progress as I did the last sprint. I achieved my goals for the Satisfactory target, and did something I believe to be equivalent to the Good target of my last report in creating the lambda calculus semantics and testing with utop. This is about as much as I expected to achieve this round.

4 Grade

I would give myself a Good grade for this phase. I clearly achieved the Satisfactory target I set in my last report, and I was able to construct and debug semantics for two forms of the lambda calculus. This is less work than the semantics for imp would have been, but I believe it was more complex so I think that evens out. As far as my plan to test my system by writing arith programs, I realized that that won't actually work until I have a lexer and parser running, so I came up with another way to test that things are working as intended. I did not manage to get to the Excellent target.

5 Goals for Final

- Satisfactory: Write the parser and lexer generators, as well as a main module which can actually run a program in the given language. Basically, finish the system at a minimal level.
- Good: Write some test programs in arith and lambda calculus to make sure the generated interpreters work. Maybe add some OUnit testing of the generated semantic modules.
- Excellent: Add polish. For example: better error reporting in the semantic translation, more complex builtin functionality, monadic io, etc.