Haskell Project Proposal

Sam Sussman-Randall & Nyx Gonzalez

October 14, 2024

1 Goals

The primary goal of this project is to have fun and learn a bit more about how a language like Haskell is built and designed. More concretely, we will be implementing a language based on System F. We'll be using the book Practical Foundations for Programming Languages (Second Edition) by Robert Harper as a guide to the more theoretical parts of this project.

The project will be broken up into a few subgoals/milestones. While it would be nice to accompolish all of them, that likely will not be practical in the three weeks we have. The milestones are listed below:

1. Basic System F
2. Additional Language Features
3. Best Effort Type Inference
4. Multi-file Linking
5. Prettier (or more interesting) Language Syntax
6. Small Basic Libraries

Note that after the first item or so, these aren't necessarily linear, just listed in the order that currently makes the most sense to us. Realistically, we're aiming to accompolish Basic System F and maybe one or two others, the rest exist so that we have options and extra goals to keep us busy incase we're overestimating the difficulty.

2 Use Cases

Our program will be able to read in a file containing a program written in our language, perform static type checking on this program, and then run it. Since we have not yet implemented the language, it's hard to give concrete examples.

3 Initial Design Outline

The basic System F implementation will start with an Abstract Syntax/Binding Tree given by the System F grammar. We can have the code for type checking and executing either all in that module, or split into separate modules. We will have another module for the parser, which converts a file/strings into the language's AST. To tie it all together we'll have a main control module. In order for the language to be actually usable, we'll need some basic primitive types that may need to be handled outside of the language itself, these can go in a separate module.

Note that the System F syntax and parser will be very bare-bones, existing mostly to make testing easier.

Most language features and other extensions will be done by having a separate AST format that compiles down to the base System F language. This is done so that we don't need to worry about maintaining type safety nearly as much while working on the extensions. This higher-level extended language will also need a parser (although it will likely just be based on and extended from the first parser), which can go in a separate module.

4 Testing

We'll be doing unit testing throughout. We'll have tests to verify that the parser is working correctly and then small program files to test that the various static/dynamic judgements hold. For the extended language we can test that each language feature/extension compiles down correctly and that programs using them work as expected.