B522 Final Paper

Ben Boskin and Paulette Koronkevich

April 28, 2018

1 Overview of project

We first chose μKanren as our DSL for the project, but hit many barriers, especially with combination of unification and streams which are found in many logic programming languages. In an effort to use an element of μKanren, we decided to use its use of fair disjunction for generating regular expressions.

In addition, in order to demonstrate our understanding of an interesting type system, we created an abstract machine for System F.

1.1 Regular Expressions

Regular expressions are a way to describe regular languages. The grammar of regular expressions that we support are (here). The forms $\cup$ and $\bullet$ require fair disjunction, which is why we chose regular expression as a simplified DSL from μKanren.

Regular expressions in programming languages are often used in print statements. For example `printf("dsjlfhdlg")` is a common method for printing (blah). It is an interesting problem to, based on the string, form a dependent function that accepts a variable number of arguments of variable types, depending on excape characters.

1.2 System F

System F is a language with polymorphic types. An abstract machine will have a more involved type system than the STLC, with types occuring in expressions. We developed an CEK-style abstract machine for System F and hopefully proved type safety. :(

2 Regular Expressions

2.1 Fair Disjunction

2.2 Dependently Typed `printf`

3 System F

3.1 Type Safety