Report

Programming III(COMP 2209)

Radoslav Radomirov Enev(rre1u17)

I. Challenges

a. Challenge 1 - *alphaNorm*

The way I did this challenge was to go through the lambda expression and determine which variables are bound and which are free, then reconstructing the lambda expression while at the same time renumbering the bound variables.

b. Challenge 2 – *countAllReds*

I ran into a slight problem with this one as I could make a functioning beta reduction function, but I could not find a way to count every reduction path possible. If there was a simple way to backtrack with haskell I could have used that. Unfortunately, I don't think I used the best approach to solve this challenge, but I couldn't think of any other way to do it.

c. Challenge 3 – *printLambda*

This challenge was very simple for me to solve, I just had to pattern match different data types and add parenthesis in strategic positions so they only appear when they should.

d. Challenge 4 – *parseLet*

Using the parsers supplied I constructed a simple parser to parse let expressions. I used the (<|>) operator from *Control.Applicative* to implement a "try" parser without having to create one from scratch myself. While I didn't create the try parser, I did make the parsers **parseLetDef**, **parseAppWithParens**, and **parseAppWithoutParens**. Along with a utility function **toInt** which turns '1' or '5' into 1 or 5 so I could avoid using the read function.

e. Challenge 5 – *letToLambda*

While I understood the algorithm outlined in the book, I could not figure out how to apply that to the tests that were given or to turn the algorithm into haskell code. I spent more than 3 weeks trying to solve this challenge, but I couldn't think of any way I could do it.

f. Challenge 6 – *lambdaToLet*

It was not very difficult to wrap my head around this conversion and came up with an algorithm in order to write the program:

- Make a **LetDef** with the first argument being the result of **makeLetStmt** (as described below) and the second argument being the result of **makeInStmt** (also described below)
- **makeLetStmt** goes through the lambda expression and pulls out data that will be needed to construct the first half of the let expression (i.e: let foo x = y)
- **makeInStmt** goes through the lambda expression and pulls out data that will be needed to constrict the second half of the let expression (i.e: … in f x)

II. Testing and Tools used.

For testing my solutions, I mostly used the supplied Tests.hs. Moreover, I have also written 3 extra tests for every challenge except for the 5th one. The tests helped me not only identify any potential flaws I had in my solutions, but also solve some of the problems I had.

The software I used to implement my solutions was Visual Studio Code. For compiling the code, I used GHCi and a windows 10 operating system. For writing this report I used Microsoft Office Word.

III. Bibliography

For this assignment I haven't used any blocks of code from other sources. Some useful materials that helped me understand the concept of solving the lamba calculus problems are:

- http://dev.stephendiehl.com/fun/lambda_calculus.html
- https://en.wikipedia.org/wiki/Lambda_calculus
- http://www.cs.nott.ac.uk/~pszgmh/pearl.pdf