Lambda Calculus Compiler

Thomas Butterwith

Aim

The aim of this project was to create a compiler capable of implementing arithmetic as well as lambda calculus (from here on referred to as λ calculus). The compiler allows a user to input a text file containing λ calculus expressions and outputs a simplified and calculated solution.
The project was designed to give an understanding of compiler design, parsing, lexing and a coherent understanding of λ calculus.

Background

Lambda Calculus
λ calculus is a way of expressing functions as formulas. It consists of a single conversion rule, variable substitution, allowing it to be easily learnt and understood but still powerful enough to create complex sequences of functions.
Expressions are defined as follows:

```
<expression>        := <id> | <function> | <application>
<function>          :=  λ <id> . <expression>
<application>       := <expression> <expression>
```

When writing λ calculus it is important to remember the variable names carry no meaning and can be easily replaced with others to increase understanding, known as alpha equivalence.

Church Numerals
In λ calculus even natural numbers can be represented as functions. By using Church Encoding it is possible to represent numbers greater than or equal to zero and apply them to functions. Numbers can be derived from the number of times a function is applied to its argument.

```
0 = λs. λz.z
1 = λs. λz.s(z)
2 = λs. λz.s(s(z)
```

Lambda Calculus Compiler

The final project is a compiler accessed through the command line to simplify λ calculus expressions, taking in a text file containing any number of λ calculus expressions, separated by a semicolon, the compiler utilises alpha-equivalence and beta simplification to output the λ expression in the simplest form possible. The program is accessed by calling the command `./lambda_compiler` with the text file passed in as the first argument after the call.
The program will then simplify the expression as far as possible and print the original statement followed by the simplified version to the command line. Any errors in syntax or unrecognised tokens within the λ expression file will be displayed in the

command line.
As this is a command line tool it is possible to pipe the output of the program into a text file by appending `> output_file_name.txt` to the end of the command should the user wish to save the output for later use.

Usage

See Usage Guide

Project References

Barendregt, H. and Barendsen, E. (1984) Introduction to Lambda Calculus. Nieuw archief voor wisenkunde 4.2 p.337-372.

Burch, C. (2012) Lambda Calulator [Online] Hendrix College. Available from: http://www.cburch.com/lambda/ [Accessed: 28th March 2015]

Jones, N. (1993) A Lambda Interpreter in ML. [Online] Department of Compuer Science, Dartmouth. Available from: http://www.cs.dartmouth.edu/~mckeeman/cs118/lectures/14.html#anchor2 [Accessed: 26th March 2015]

Minsky, Y., Madhavapeddy, A., and Hickey, J. (2013). Real World OCaml: functional programming for the masses. O'Reilly Media, Inc..

Mogensen, T. Æ. (2010) Basics of Compiler Design. Anniversary Ed. Department of Computer Science, University of Copenhagen

Ocaml (Unknown) 99 Problems (solved) in Ocaml. [Online] Ocaml.org. Available from: https://ocaml.org/learn/tutorials/99problems.html [Accessed: October 2014]

Rojas, R. A Tutorial Introduction to the Lambda Calculus. [Online] University of Texas Dallas. Available from : http://www. utdallas. edu/~ gupta/courses/apl/lambda. pdf. [Accessed: December 2014]

Selinger, P. (2008) Lecture Notes on the Lambda Calculus. [Online] Department of Mathematics and Statistics Dalhousie University, Halifax, Canada. Available from : http://cs.simons-rock.edu/cmpt320/selinger.pdf [Accessed: December 2014]

Smith, J. B. (2007) Practical Ocaml. Apress.

SooHyoung, O. (2004) Ocamlyacc Tutorial. [Online] Programming Languages Laboratory, Korea Advanced Institute of Science and Technology. Available from: http://plus.kaist.ac.kr/~shoh/ocaml/ocamllex-ocamlyacc/ocamlyacc-tutorial/ [Accessed: 20th January 2015]