Masaryk University

Faculty of Informatics

Online Lambda Calculus Evaluator

Bachelor Thesis

Jakub Kadlecaj

Spring 2018

DRAFT

Contents

0 Introduction

Lambda calculus is a formal system of logic, concerned with functions described as substitution-based evaluation rules. It plays an important role in computer science and mathematics, with applications in other fields, such as linguistics and philosophy.

The aim of this thesis is to design and implement a web application that evaluates user-input lambda calculus terms, both pure and typed, with a possibility of selecting a particular evaluation strategy and step-by-step evaluation, multiple importing and exporting options, and several other features lacking in publicly available pre-existing solutions. The application should be useful to students of the course *IA014 – Advanced Functional Programming*, as it is intended to aid in the understanding of fundamental principles of lambda calculus taught therein.

Firstly, a general overview of the importance of λ-calculus will be presented, including, to some extent overlapping topics of historical significance and real-world applications—especially usage in programming languages, then a rigorous definition of lambda calculus' syntax and semantics, together with descriptions of the implemented evaluation strategies, type systems, and extensions will be provided, and lastly, a thorough description of the evaluator application itself will be given. In the appendix, one can find examples of some of the possible inputs to the application, together with more nuanced and comprehensive characterisation of the evaluator, potentially serving as a reference manual.

1 Background

1.1 Historical Significance

kleene rosser paradox
    computability
    turing machinas
    montague and linguistics and TIL

1.2 Real-World Applications

lambda funcs in modern langs
    robert milner, ML
    huskel
    lisp

2 Lambda Calculus

2.1 Pure Lambda Calculus

2.1.1 Term Syntax

The language of λ-calculus is generated by the following grammar in Backus-Naur form, consisting only of three rules:

$$
\begin{array}{lcl}
\langle term \rangle & ::= & \langle variable \rangle \\
 & \mid & (\lambda \langle variabe \rangle . \langle term \rangle) \\
 & \mid & (\langle term \rangle \ \langle term \rangle)
\end{array}
$$

Generally, a *variable* could be any unambiguous identifier that is a member of a chosen countable set, but in this work, the set of variables is restricted to lowercase characters optionally followed by a numerical subscript, as the general case could interfere with the syntactic conventions of which the definition is given in the following section.

2.1.2 Syntactic Conventions

To make terms more succinct and convenient to write and read, the following syntactic conventions are being used:

$$
\begin{aligned}
\lambda x_1 x_2 x_3 \dots x_n . M &\equiv (\lambda x_1 . (\lambda x_2 . (\lambda x_3 . (\dots (\lambda x_n . M) \dots)))) \\
M_1 \, M_2 \, M_3 \dots M_n &\equiv (\dots ((M_1 \, M_2) \, M_3) \dots M_n) \\
\lambda x . x_1 \, x_2 \dots x_n &\equiv (\lambda x . (x_1 \, x_2 \dots x_n))
\end{aligned}
$$

The intuition behind the first two rules is rather straighforward, as the rules seemingly emulate process of *currying*, i.e. translation between function of multiple arguments *(left-hand side)* and multiple functions of a single argument *(right-hand side)*. Clearly, this is only a matter of syntactic simplicity and has no effect on the actual semantics. The third rule necessitates a precedence of the application rule over the abstraction rule.

Terms can also be given a name—the choice of term names, or *aliases*, is restricted to strings of uppercase letters, and strings of numerals. The means of assigning terms to numeral names is described in section *Church Encoding*.

2.1.3 Variable Binding and Substitution

The substitution of one term for another lies at the very heart of λ-calculus' evaluation mechanics. In order to define reductions and reduction strategies later, the concept of free and bound variables must be introduced first.

$$\begin{aligned}\mathrm{FV}(x) &= \{x\}\\ \mathrm{FV}(M\,N) &= \mathrm{FV}(M) \cup \mathrm{FV}(N)\\ \mathrm{FV}(\lambda x.M) &= \mathrm{FV}(M) \setminus \{x\}\end{aligned}$$

where x is some variable, and M and N are terms

For example, $\mathrm{FV}(\lambda xyz.x) = \{\}$, $\mathrm{FV}(\lambda e.y\ e) = \{y\}$, and $\mathrm{FV}\big((\lambda w.w)\ w\big) = \{w\}$. Some variable x occuring in a term M is said to be bound, if it is not a member of the set of free variables of the term M.

Although apparently trivial, the substitution of free variables of some term with an another term, the result being written as $M[x := N]$, can be precarious, for some of the free variables of term N can become bound in the resulting term. A simple illustration of this phenomenon is the following function, $\lambda xy.x$, that could be understood as a function taking two parameters and returning the first. When this function is partially applied to a particular argument, $(\lambda xy.x)\,y$, naïve replacement of bound variable x for argument y would yield $y.y$, that is, the identity function, a very different result from the expected function returning the first, already applied parameter y, for any input. The solution to this problem leads to renaming variables in the process of α-conversion.

$$x[x := N] \equiv N \tag{1}$$

$$y[x := N] \equiv N,\ \ \mathit{where}\ x \neq y \tag{2}$$

$$(M_1\,M_2)[x := N] \equiv (M_1[x := N])\,(M_2[x := N]) \tag{3}$$

$$(\lambda x.M)[x := N] \equiv \lambda x.M \tag{4}$$

$$(\lambda y.M)[x := N] \equiv \lambda y.(M[x := N]),\ \ \mathit{where}\ x \neq y \tag{5}$$

2.1.4 Reductions and Reduction Strategies

reduction strats church rosser

2.1.5 Church Encoding

2.1.6 Recursion

2.1.7 Implemented Extensions and Syntactic Sugar

higher order funs

2.2 Type Systems

2.2.1 Simply-Typed

2.2.2 Hindley-Milner type system

3 Evaluator

3.1 Technical Aspects

javascript lol

3.2 Features

info bar
  help page
  compability table

3.2.1 User's Input and Parsing

parser gramamr

3.2.2 Alias management

3.2.3 Display Settings

alias expansion and shorthand

3.2.4 Evaluation

clicking and stuff

3.2.5 Input & Output

using current display settings printing
  latex
  files, url printing

3.3 Comparision with Existing Solutions

4 Conclusion

It's done, thank you.

A Examples and Reference Manual