Introduction to Lambda Calculus

Maciek Makowski

April 26, 2014

1 Motivation

TODO

2 Syntax

$$\begin{array}{lcl} \langle \textit{term} \rangle & ::= & v \\ & | & \lambda v.\langle \textit{term} \rangle \\ & | & \langle \textit{term} \rangle\ \langle \textit{term} \rangle \end{array}$$

where $v \in V$, a set of variable names.

Examples: v_1, $x\ y$, $(\lambda a.\lambda b.a)\ c\ (\lambda a.b)$

The rule with

3 Rewriting Rules

The grammar tells us how to generate arbitrary lambda-terms. We can define the following operations on those terms:

- renaming of bound variables (α-*conversion*); e.g. $(\lambda x.x\,y)\ (\lambda x.x) \longleftrightarrow_\alpha (\lambda a.a\,y)\ (\lambda b.b)$
- removal of abstraction under application (β-*reduction*); e.g. $(\lambda x.x\,y)\ (\lambda x.x) \longrightarrow_\beta y$
- introduction/removal of redundant abstraction (η-*conversion*); e.g. $\lambda x.y\,x \longleftrightarrow_\eta y$

TODO: note on subtleties in substitution

TODO: expand on each rule

4 Foundational Theory

What exactly is the number 2? Foundational theories of mathematics attempt to provide a concrete answer in terms of some primitive objects. The best known example is that based in set theory. Using Peano's arithmetic where $2 = S(S(0))$ (S is the successor function) natural numbers can be modelled as follows:

$$\begin{aligned}0 &= \emptyset\\ 1 &= \{\emptyset\}\\ 2 &= \{\{\emptyset\},\emptyset\}\end{aligned}$$

$\boxed{\text{In general, } S(n) = n \cup \{n\}.}$

It turns out that all generally known mathematical concepts can be modelled in set theory in some way. Lambda calculus was conceived by Church as an alternative foundational theory in which mathematics can be embedded[2]. How do we model natural numbers in it?

$$\begin{aligned}0 &= \lambda s.\lambda z.z\\ 1 &= \lambda s.\lambda z.s\,z\\ 2 &= \lambda s.\lambda z.s\,(s\,z)\end{aligned}$$

$\boxed{\text{In general, } S(n) = \lambda s.\lambda z\, \underbrace{s\,(\ldots s\,(s}_{n}\ z)\ldots).}$

5 Programming Language

TODO

6 Model of Computation

TODO

7 Church-Rosser Theorem

TODO

8 Curry-Howard Correspondence

TODO

9 Further Reading

A direct inspiration for this talk was the presentation of lambda calculus in [1]. The book is very well written and builds a sophisticated type system in easy to follow steps, starting from untyped lambda calculus.

TODO

References

[1] Benjamin C. Pierce, *Types and Programming Languages*, `http://www.cis.upenn.edu/~bcpierce/tapl/`

[2] Klaus Grue, *Lambda calculus as a foundation of mathematics*, `http://www.diku.dk/~grue/papers/church/church.html`

[3] Henk Berendregt, Erik Barendsen, *Introduction to Lambda Calculus*, `http://www.cse.chalmers.se/research/group/logic/TypesSS05/Extra/geuvers.pdf`