Lambda Calculus

Nicholas Miehlbradt

October 28, 2022

Syntax

Lambda calculus is recursively defined.

$$\begin{aligned} E &:= x \\ &:= \lambda x.E \\ &:= E\ E \end{aligned}$$

Here x can be any name and Es on the right hand side can be replaced by any sub expression constructed using the same rules.

Binding

Without brackets, these are the binding rules:

$$\lambda x.f\ a\ b = (\lambda x.f\ a\ b)$$

$$a\ b\ c = (a\ b)\ c$$

If we want to express something different, we use brackets.

Free variables

A variable is free if it is not bound by a lambda e.g.

x is bound: $\lambda x.x$

y is free: $\lambda x.y$

$$\lambda x.y\ (\lambda y.z\ y\ x)$$

α-Conversion

What letter we use after our λs doesn't matter e.g.

$$\lambda x.x$$
$$\lambda y.y$$

are the same function (they do the same thing)

We can change the name of a variable after a λ as long as we change all places where it would be substituted. This is called α-conversion.

β-Reduction

Whenever we have an expression of the form:

$$(\lambda x.E)F$$

We can replace it with E where we replace all occurences of x in E with F e.g.

$$\begin{aligned}&(\lambda x.\lambda y.x\ y)\ (\lambda z.z)\\ \rightarrow&\lambda y.(\lambda z.z)\ y\end{aligned}$$

Representing Data

How can we use this to represent data?

Booleans

$$True := \lambda x.\lambda y.x$$

$$False := \lambda x.\lambda y.y$$

Numbers

This method of representing numbers is called Church numerals.

$$0 := \lambda f.\lambda x.x$$
$$1 := \lambda f.\lambda x.f\ x$$
$$2 := \lambda f.\lambda x.f\ (f\ x)$$
$$3 := \lambda f.\lambda x.f\ (f\ (f\ x))$$