

Notation

Three kinds of λ -terms:

1. variables: x, y, z, \dots
2. application (function call): MN
3. abstraction (function definition): $\lambda x.M$

Free Variables:

$$\begin{aligned} FV x &= \{x\} \\ FV (\lambda x.M) &= (FV M) - \{x\} \\ FV (MN) &= (FV M) \cup (FV N) \end{aligned}$$

Theory

1. α -equivalence, α -renaming

$$\lambda x.M =_{\alpha} \lambda y.[y/x]M$$

2. β -reduction

$$(\lambda x.M)N \rightarrow_{\beta} [N/x]M$$

3. β -equivalence

$$\begin{aligned} (\lambda x.M)N &=_{\beta} [N/x]M \\ [N/x]M &=_{\beta} (\lambda x.M)N \end{aligned}$$

4. η -equivalence (not very important)

$$M =_{\eta} \lambda x.Mx$$

$[N/x]M$ represents the substitution of N for all of the free occurrences of x in M . These rules may be applied to individual subterms of a larger term.

Terminology/Concepts

Church-Rosser Theorem (confluence)

$$M =_{\beta} N \Rightarrow \exists Z [M \rightarrow_{\beta} \dots \rightarrow_{\beta} Z \wedge N \rightarrow_{\beta} \dots \rightarrow_{\beta} Z]$$

normal form result of reducing a λ -term until it cannot be reduced anymore

currying simulating multiple arguments with a higher-order function that takes one argument and returns a function that takes the next argument

Important Functions

1. Identity

$$\mathbf{I} \equiv \lambda x.x$$

2. Functions Important to Computability

(a) fixed point combinator, \mathbf{Y} (builds recursive functions)

$$\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

$$\forall F.YF = F(YF)$$

(b) alternative call-by-value fixed point combinator, \mathbf{Z} (usable in Scheme)

$$\mathbf{Z} \equiv \lambda f.(\lambda x.f(\lambda y.(xx)y))(\lambda x.f(\lambda y.(xx)y)) =_{\eta} \mathbf{Y}$$

$$\forall F.ZF = F(\lambda y.(ZF)y) =_{\eta} F(ZF)$$

(c) Omega, never terminates (undefined)

$$\mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$$

3. Church Booleans

$$\mathbf{T} \equiv \lambda x.\lambda y.x$$

$$\mathbf{F} \equiv \lambda x.\lambda y.y$$

$$\mathbf{if} \equiv \lambda p.\lambda c.\lambda a.(pca)$$

$$\mathbf{and} \equiv \lambda x.\lambda y.(xy)x$$

$$\mathbf{or} \equiv \lambda x.\lambda y.(xx)y$$

4. Church Numerals

$$\mathbf{0} \equiv \lambda f.\lambda x.x$$

$$\mathbf{1} \equiv \lambda f.\lambda x.fx$$

$$\mathbf{2} \equiv \lambda f.\lambda x.f(fx)$$

$$\mathbf{3} \equiv \lambda f.\lambda x.f(f(fx))$$

$$\mathbf{n} \equiv \lambda f.\lambda x.f^n(x)$$

$$\mathbf{S}^+ \equiv \lambda n.\lambda f.\lambda x.f((nf)x)$$

$$\mathbf{P}^- \equiv \lambda n.\lambda f.\lambda x.n(\lambda g.\lambda h.h(gf))(\lambda u.x)(\lambda u.u)$$

$$+ \equiv \lambda m\lambda n.\lambda f.\lambda x.(mf)((nf)x)$$

$$\times \equiv \lambda m\lambda n.\lambda f.\lambda x.(m(nf))x$$

Sources: Mitchell, Concepts in Programming Languages; Hankin, An Introduction to Lambda Calculi for Computer Scientists; Wikipedia, Church encoding; Wikipedia, Fixed point combinator