

1 Syntax of lambda calculus

Assume a countable set of variable names, denoted by $a, b, c, \dots, x, y, z, a_0, a_1, \dots$

Definition. A λ -term is defined by the following context-free grammar:

$$\begin{aligned} \langle term \rangle &:= \langle name \rangle \\ &| (\lambda \langle name \rangle . \langle term \rangle) \\ &| (\langle term \rangle \langle term \rangle) \end{aligned}$$

Conventions.

1. **Function application** is left-associative, so $((A_1 A_2) A_3) \dots A_k$ can be abbreviated as $A_1 A_2 A_3 \dots A_k$
2. Nested **abstractions** $(\lambda x_1. (\lambda x_2. (\dots \lambda x_k. A) \dots))$ can be abbreviated as $\lambda x_1 x_2 \dots x_k. A$

Example.

$\lambda xy. FAB$ means $((\lambda x. (\lambda y. F)) A) B$

2 Free variables

1. $\langle name \rangle$ is free in $\langle name \rangle$
2. $\langle name \rangle$ is free in $\lambda \langle name' \rangle . \langle term \rangle$ if $\langle name \rangle \neq \langle name' \rangle$ and $\langle name \rangle$ is free in $\langle term \rangle$
3. $\langle name \rangle$ is free in $\langle term' \rangle \langle term'' \rangle$ if $\langle name \rangle$ is free in $\langle term' \rangle$ or $\langle name \rangle$ is free in $\langle term'' \rangle$

3 Bound variables

1. $\langle name \rangle$ is bound in $\lambda \langle name' \rangle . \langle term \rangle$ if $\langle name \rangle = \langle name' \rangle$ or $\langle name \rangle$ is bound in $\langle term \rangle$
2. $\langle name \rangle$ is bound in $\langle term' \rangle \langle term'' \rangle$ if $\langle name \rangle$ is bound in $\langle term' \rangle$ or $\langle name \rangle$ is bound in $\langle term'' \rangle$

Examples. Free and bound occurrences of variables:

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

4 Reductions

Denote by $T[x := E]$ the term in which every free occurrence of x is replaced by E .

α -conversion: Bound variables can be renamed, $\lambda x. F \equiv \lambda t. F[x := t]$, provided

that t is not free in F and is not bound by a λ in F whenever it replaces an x .

β -reduction: Applying a function to the argument: $(\lambda x.F)A \equiv F[x := A]$ provided that all free occurrences in A remain free in $F[x := A]$.

Definition. A λ -term is in a normal form if no β -reduction can be applied to it.

Theorem. If a λ -term has a normal form, then the normal form is unique (up to renaming of bound variables).

Computing the normal form: Keep on replacing the leftmost bound variable by the actual argument until no further reduction is possible.

This does not terminate if and only if the initial term has no normal form.

5 Church numerals

The **Church numerals** C_0, C_1, \dots are defined as:

$$\begin{array}{lll} C_0 & \equiv & \lambda sz.z \\ C_1 & \equiv & \lambda sz.s(z) \\ C_2 & \equiv & \lambda sz.s(s(z)) \\ \dots & \dots & \dots \end{array}$$

The successor can be defined as:

$$S \equiv \lambda uvw.v(uvw)$$

Exercise. Verify that $SC_n = C_{n+1}$ for every $n \in \mathbb{N}$

Lemma. For every two terms F and A , $C_n FA = F^{(n)} A$

Corollary. Doing addition in λ -calculus: $C_n SC_m = C_{n+m}$

6 Predecessor

Define *True* and *False* by:

$$\begin{array}{lll} T & \equiv & \lambda xy.x \\ F & \equiv & \lambda xy.y \end{array}$$

Represent a pair (a, b) by $\lambda z.zab$. Define:

$$\Phi = \lambda pz.z(S(pT))(pT)$$

The **predecessor** is defined as:

$$P \equiv \lambda n.n\Phi(\lambda z.zC_0C_0)F$$