The Untyped Lambda Calculus

Alonzo Church (1903 - 1995)

λ-calculus: syntax

Grammar for terms:

$$
\begin{array}{lllll}
t, u, v ::= & x & \text{(variable)} & | \\
& t\ u & \text{(application)} & | \\
& \lambda x.t & \text{(abstraction)} &
\end{array}
$$

Notation :

Application is left-associative so that $t_1\ t_2\ \ldots t_n$ means $(\ldots(t_1\ t_2)\ \ldots t_n)$.

$\lambda x_1 \ldots x_n.t$ means $\lambda x_1.\ldots.\lambda x_n.t$.

Inductive definition of λ-terms

$$\frac{x \text{ is a variable}}{x \text{ is a } \lambda\text{-term}} \text{ (Var)}$$

$$\frac{t \text{ is a } \lambda\text{-term} \qquad u \text{ is a } \lambda\text{-term}}{t\ u \text{ is a } \lambda\text{-term}} \text{ (App)}$$

$$\frac{t \text{ is a } \lambda\text{-term} \qquad x \text{ is a variable}}{\lambda x.t \text{ is a } \lambda\text{-term}} \text{ (Lamb)}$$

Size of λ-terms

- $size(x) := 1$
- $size(uv) := 1 + size(u) + size(v)$
- $size(\lambda x.u) := 1 + size(u)$

Free and bound variables

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

The set of free and bound variables are defined as follows

$$\begin{array}{lcllcl}
\mathtt{fv}(x) & := & \{x\} & \mathtt{bv}(x) & := & \emptyset \\
\mathtt{fv}(t\ u) & := & \mathtt{fv}(t) \cup \mathtt{fv}(u) & \mathtt{bv}(t\ u) & := & \mathtt{bv}(t) \cup \mathtt{bv}(u) \\
\mathtt{fv}(\lambda x.t) & := & \mathtt{fv}(t) \setminus \{x\} & \mathtt{bv}(\lambda x.t) & := & \mathtt{bv}(t) \cup \{x\}
\end{array}$$

But for $t = x\ (\lambda x.x)$

$$\mathtt{fv}(x\ (\lambda x.x)) = \{x\} = \mathtt{bv}(x\ (\lambda x.x))$$

A term is closed iff $\mathtt{fv}(t) = \emptyset$.

Alpha-conversion

The relation $=_\alpha$, usually called renaming or alpha-conversion, is inductively defined as:

$$\frac{y \text{ is a fresh variable}}{\lambda x.t =_\alpha \lambda y.t[x \backslash\backslash y]} \qquad \frac{t =_\alpha t'}{\lambda x.t =_\alpha \lambda x.t'} \qquad \frac{t =_\alpha t' \qquad u =_\alpha u'}{t\ u =_\alpha t'\ u'}$$

where the operation $t[x \backslash\backslash y]$ denotes the replacement of all the free occurrences of x in t by a fresh variable y. Formally,

$$\begin{array}{lcll}
x[x \backslash\backslash y] & := & y & \\
z[x \backslash\backslash y] & := & z & x \neq z \\
(t\ u)[x \backslash\backslash y] & := & t[x \backslash\backslash y]\ u[x \backslash\backslash y] & \\
(\lambda x.t)[x \backslash\backslash y] & := & \lambda x.t & \\
(\lambda z.t)[x \backslash\backslash y] & := & \lambda z.t[x \backslash\backslash y] & x \neq z
\end{array}$$

Examples

$$\lambda x.(\lambda x.x\ z) =_\alpha \lambda y.(\lambda x.x\ z) =_\alpha \lambda y.(\lambda y.y\ z)$$

$$x\ (\lambda x.x) =_\alpha x\ (\lambda z.z)$$

Barendregt variable convention

From now on we assume the following variable convention:

1. No variable is both free and bound.
2. Bound variables have all different names.

Example

$x\ (\lambda z.z)$ is OK, $x\ (\lambda x.x)$ is not OK, $\lambda x.\lambda y.x\ z$ is OK but $\lambda x.\lambda x.x\ z$ is not OK.

Theorem

For every λ-term t there is λ-term u verifying the Barendregt convention such that $t =_\alpha u$.

Indeed, $x\ (\lambda x.x) =_\alpha x\ (\lambda z.z)$ and $\lambda x.\lambda x.x\ z =_\alpha \lambda y.\lambda x.x\ z$.

Operational semantics of λ-calculus

A one-step β-reduction is given inductively by

$$\frac{}{(\lambda x.t)\ u \to_\beta\ t\{x \backslash u\}} \qquad \frac{t \to_\beta\ t'}{\lambda x.t \to_\beta\ \lambda x.t'}$$

$$\frac{t \to_\beta\ t'}{t\ u \to_\beta\ t'\ u} \qquad \frac{u \to_\beta\ u'}{t\ u \to_\beta\ t\ u'}$$

What is exactly $\_\{\_\backslash\_\}$?

Towards a notion of substitution: warning!

$$(\lambda x.(\lambda y.x))\ y \rightarrow_\beta\ (\lambda y.x)\{x \backslash y\} = \lambda y.y$$

Incorrect

$$(\lambda x.(\lambda y.x))\ y =_\alpha (\lambda x.(\lambda z.x))\ y \rightarrow_\beta\ (\lambda z.x)\{x \backslash y\} = \lambda z.y$$

Correct

A simple notion of higher-order substitution

$t\{x/u\}$ means replace all the free occurrences of x in t by u.
This operation is defined modulo α-conversion as follows:

$$
\begin{array}{lcll}
x\{x/u\} & := & u & \\
y\{x/u\} & := & y & \text{if } x \neq y \\
(\lambda y.v)\{x/u\} & := & \lambda y.v\{x/u\} & \text{if } x \neq y \text{ and } y \notin \mathtt{fv}(u) \text{ (no capture holds)} \\
(t\ v)\{x/u\} & := & (t\{x/u\}\ v\{x/u\}) &
\end{array}
$$

A terminating β-reduction sequences

$$
\begin{array}{ll}
(\lambda x.\lambda f.x\ f\ y)\ (\lambda z.z)\ (\lambda w.w\ w) & \rightarrow_\beta \\
(\lambda f.x\ f\ y)\ \{x \backslash \lambda z.z\}\ (\lambda w.w\ w) & = \\
(\lambda f.(\lambda z.z)\ f\ y)\ (\lambda w.w\ w) & \rightarrow_\beta \\
(\lambda f.f\ y)\ (\lambda w.w\ w) & \rightarrow_\beta \\
(f\ y)\{f \backslash \lambda w.w\ w\} & = \\
(\lambda w.w\ w)\ y & \rightarrow_\beta \\
(w\ w)\ \{w \backslash y\} & = \\
(y\ y) &
\end{array}
$$

A non terminating β-reduction sequences

Let $\Delta_f = \lambda x.f\ (x\ x)$.

$$
\begin{array}{ll}
\Delta_f\ \Delta_f & \rightarrow_\beta \\
(f\ (x\ x))\{x \backslash \Delta_f\} & = \\
f\ (\Delta_f\ \Delta_f) & \rightarrow_\beta \\
f\ (f\ (x\ x))\{x \backslash \Delta_f\} & = \\
f\ (f\ (\Delta_f\ \Delta_f)) & \rightarrow_\beta \\
\vdots &
\end{array}
$$

Expressivity

- Arithmetic.
- Booleans.
- Recursion.
- The λ-calculus is Turing complete.

Properties of λ-calculus

[Free variables decrease] If $t \to_\beta t'$, then $fv(t) \supseteq fv(t')$.

Example

In $t_0 = (\lambda x.x)(\lambda z.zz) \to_\beta \lambda z.zz = t_1$ we have $\emptyset = \mathtt{fv}(t_0) = \mathtt{fv}(t_1) = \emptyset$.
In $t_0 = (\lambda x.y)(zz) \to_\beta y = t_1$ we have $\{y, z\} = \mathtt{fv}(t_0) \supseteq \mathtt{fv}(t_1) = \{y\}$.

Properties of λ-calculus

[Confluence] The reduction relation $\rightarrow_\beta$ is confluent.

Proof.

By Tait-Martin-Loëf technique (on blackboard). □