Lambda Calculus

Alonzo Church (1903 - 1995)

λ-calculus : syntax

Grammar for terms :

$$t, u ::= \quad x \quad \text{(variable)} \quad | \qquad (t\ u) \quad \text{(application)} \quad | \qquad \lambda x.t \quad \text{(abstraction)}$$

Notation :

$t_1\ t_2\ \ldots t_n$ 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 for λ-terms

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

$$\frac{t \text{ and } u \text{ are } \lambda\text{-terms}}{(t\ u) \text{ is a } \lambda\text{-term}} \text{ (App)} \qquad \frac{t \text{ is a } \lambda\text{-term}}{\lambda x.t \text{ is a } \lambda\text{-term}} \text{ (Lamb)}$$

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

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

Alpha-conversion

The congruence generated by the following axiom :

$$\lambda x.t =_\alpha \lambda y.t[x/\!\!/y] \text{ for } y \text{ fresh}$$

where $t[x/\!\!/y]$ means the replacement of all the free occurrences of x in t by a fresh variable y.

Defining the notion of fresh replacement

$$
\begin{array}{lcl}
x[x/\!\!/y] & = & y \\
z[x/\!\!/y] & = & z \\
(t\ u)[x/\!\!/y] & = & (t[x/\!\!/y]\ u[x/\!\!/y]) \\
(\lambda x.t)[x/\!\!/y] & = & \lambda x.t \\
(\lambda z.t)[x/\!\!/y] & = & \lambda z.t[x/\!\!/y]
\end{array}
$$

Example :

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

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

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

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

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

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

Towards a notion of substitution : warning !

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

Incorrect

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

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 :

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

$y \notin \mathtt{fv}(u)$

A terminating β-reduction sequences

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

A non terminating β-reduction sequences

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

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

Properties of λ-calculus

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

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

Curry-Howard Isomorphism

Logical system	$\Leftrightarrow$	**Language**
	$\Leftrightarrow$	Types
	$\Leftrightarrow$	Programs
Proof normalisation	$\Leftrightarrow$	Program Evaluation

also called cut elimination

Curry'58, Howard'68

Natural deduction as typed λ-calculus

$$\frac{}{\Gamma, x : A \vdash x : A} \quad (ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \quad (\to i) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : B} \quad (\to e)$$

We denote by $\Gamma \vdash t : A$ the derivability/typing relation.

Typed Properties

[Subject Reduction] If $\Gamma \vdash t : A$ and $t \rightarrow_\beta t'$, then $\Gamma \vdash t' : A$.

[Strong Normalization] Every typed term is normalising : if $\Gamma \vdash t : A$, then $t \in SN_\beta$.

Some General Remarks

- SN is not stable by substitution. Example : $x\ x \in SN$, $\lambda y.y\ y \in SN$, but $(x\ x)\{x/\lambda y.y\ y\} = \Delta\ \Delta \notin SN$.
- When using typed terms $t\{x/u\}$ means that x and u have the same type.
- $t \in SN$

  iff there is no infinite reduction sequence starting at t

  iff every reduction sequence starting at t is finite

  iff $\forall t'\ (t \rightarrow_\beta\ t')$ implies $t' \in SN$.
- A particular case is $t \in SN$ if t is in normal form.
- The standard order between types is given by $A < A \rightarrow B$ and $B < A \rightarrow B$.

  Thus base types are minimal with respect this order.

- $u \in SN$ iff $\lambda y.u \in SN$.
- $u_1 \ldots n_n \in SN$ iff $x\ u_1 \ldots u_n \in SN$.
- Given $t \in SN$ we define $\mu(t)$ as the maximal lenght of a reduction sequence starting at t. We observe that $t \rightarrow\ t'$ implies $\mu(t') < \mu(t)$.

First Proof of the SN property (i)

Définition : Let M be of type $A_1 \to \ldots \to \ldots A_n \to \tau$.
$M \in SC$ iff $\forall R_i : A_i \in SC \; M \; R_1 \ldots R_n \in SN$

The definition implies

1. $SC \subseteq SN$.
2. SC is closed under β.
3. $x \in SC$ for every variable x (using 1).

First Proof of the SN property (ii)

Lemme : Every typed term is SN.

Proof. We prove forall typed M and all type preserving σ mapping free variables of M to terms in SC, $M\sigma \in SC$. We proceed by induction on the typed term M. The theorem then follows by setting σ to the identity.

- If $M = x$, then $x\sigma = \sigma(x) \in SC$ by hypothesis.
- If $M = NL$, then let $\vec{R_n} \in SC$. We have $N\sigma$ and $L\sigma$ in SC by i.h. Then $(N\ L)\sigma\vec{R_n} = N\sigma\ L\sigma\vec{R_n} \in SN$ by definition.

- If $M = \lambda x.N$, then $(\lambda x.N)\sigma = \lambda x.N\sigma$. By the i.h. $N(\sigma \cup \{x/x\}) = N\sigma \in SC \subseteq SN$. To show $\lambda x.N\sigma \in SC$ we show $(\lambda x.N\sigma)P_1 \ldots P_n \in SN$ for $P_1 \ldots P_n \in SC \subseteq SN$. For that, it is sufficient to show that all its reducts are in SN. We reason by induction on $\mu(N\sigma) + \Sigma_i\ \mu(P_i)$. The reducts are
  - $(\lambda x.N')P_1 \ldots P_n$, where $N\sigma \rightarrow\ N'$. We apply the i.h.
  - $(\lambda x.N\sigma)P_1 \ldots P'_i \ldots P_n$, where $P_i \rightarrow\ P'_i$. We apply the i.h.
  - $N\sigma\{x/P_1\}P_2 \ldots P_n$. The term $N\sigma\{x/P_1\}$ is in SC by the i.h. and thus $N\sigma\{x/P_1\}P_2 \ldots P_n \in SN$ by definition.

∎

Second proof of the SN property

1. Define SN inductively :
   - $M_1, \ldots, M_n \in SN$ implies $x\ M_1\ \ldots M_n \in SN$.
   - $M \in SN$ implies $\lambda x.M \in SN$.
   - $M\{x/N\}\vec{P_n} \in SN$ and $N \in SN$ implies $(\lambda x.M)\ N\ \vec{P_n} \in SN$.
2. Define Λ_A inductively :
   - If x is a variable of type A, then $x \in \Lambda_A$.
   - If $t \in \Lambda_C$ and x is a variable of type B, then $\lambda x.t \in \Lambda_{B \to C}$.
   - If $t \in \Lambda_{B \to A}$ and $u \in \Lambda_B$, then $t\ u \in \Lambda_A$.
3. Define $SN_A = \{M \mid SN \cap \Lambda_A\}$.
4. Define $X \to Y = \{M \mid \forall N \in X\ (MN) \in Y\}$

5. Show $SN_\beta = SN$.
6. Show $\Lambda_{A \to B} = \Lambda_A \to \Lambda_B$
7. Show $SN_A \to SN_B \subseteq SN_{A \to B}$ (easy).
8. If $N \in SN_{A_1} \to SN_{A_2} \to \ldots \to SN_{A_m}$ with A_m of base type and $P \in SN_B$, then $P\{x/N\} \in SN_B$. (induction on SN using 7).
9. Show $SN_{A \to B} \subseteq SN_A \to SN_B$ (using 8).
10. Show that $\Lambda_A \subseteq SN_A$ (by induction using 9).

Third Proof of the SN property

Lemme : If t and u are typed and SN, then $t\{x/u\}$ is SN.

Proof. By induction on $\langle type(u), \mu(t), size(t)\rangle$.

- The base case $\langle$ base type, 0, 1 $\rangle$ is trivial.
- Case $t = \lambda y.v$ is straightforward ($size(t)$ strictly decreases).
- Case $t = y\ \vec{c_n}$ with $x \neq y$ is straightforward ($\mu(t)$ decreases and $size(t)$ strictly decreases.).
- Case $t = x$. We have $x\{x/u\} = u \in SN$ by hypothesis.
- Case $t = x\ b\ \vec{c_n}$. By i.h. $B = b\{x/u\}$ and $C_i = c_i\{x/u\}$ are SN. We want to show that $u\ B\ \vec{C_n}$ is SN. It is sufficient to show that all its reducts are SN. We reason by induction on $\mu(u) + \mu(B) + \Sigma_i\ \mu(C_i)$. The reducts are

- $u'\ B\ \vec{C_n}$, where $u \rightarrow\ u'$. Apply the i.h.
- $u'\ B'\ \vec{C_n}$, where $B \rightarrow\ B'$. Apply the i.h.
- $u'\ B\ C_1 \ldots C_i' \ldots C_n$, where $C_i \rightarrow\ C_i'$. Apply the i.h.
- $u'\{y/B\}\vec{C_n}$, where $u = \lambda y.u'$. But
  $u'\{y/B\}\vec{C_n} = u'\{y/b\}\vec{c_n}\{x/u\}$ and
  $\mu(u'\{y/b\}\vec{c_n}) < \mu((\lambda y.u')\ b\ \vec{c_n})$. We thus conclude by the i.h.

- Case $t = (\lambda z.b)\ c\ \vec{d}$. By i.h. $B = b\{x/u\}$ and $C = c\{x/u\}$ and $D_i = d_i\{x/u\}$ are SN. Suppose $t\{x/u\} = (\lambda z.B)\ C\ \vec{D_n} \notin SN$. Then $B\{z/C\}\ \vec{D_n} \notin SN$. But $B\{z/C\}\ \vec{D_n} = (b\{z/c\}\vec{d_n})\{x/u\}$ and $\mu(b\{z/c\}\vec{d_n}) < \mu(t)$. Thus $B\{z/C\}\ \vec{D_n} \in SN$ by the i.h. Contradiction. Thus $t\{x/u\} = (\lambda z.B)\ C\ \vec{D_n} \in SN$.

∎

Theorem : If t is typable, then t is SN.

Proof. By induction on the structure of t.

- Case $t = x$ is trivial.
- Case $t = \lambda y.u$ holds by the i.h.
- For Case $t = (u\ v)$ use the fact that $t = (z\ v)\{z/u\}$ and apply previous lemma.

∎

Fourth proof of the SN property

See for example

Alexandre Miquel.

A combinatorial proof of strong normalisation for the simply typed lambda-calculus.

`http://www.pps.jussieu.fr/~miquel/publis/snlam.pdf`