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

$$(\lambda x.t)\ u \rightarrow_\beta\ t\{x/u\}$$

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 $t \in SN_\beta$.

First Proof of the SN property (by V. van Oostrom)

Définition : $M \in SC if f \forall \vec{R} \in SC\ M\ \vec{R} \in \Lambda is SN$

From the definition SC implies SN, SC is closed under β and SC holds for every variable x, since $\vec{R}$ in SC, then $\vec{R}$ in SN, then $x\ \vec{R}$ in SN.

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 is a variable, then $x\sigma = \sigma(x)$ is SC by hypothesis.
- If $M = NL$, then let $\vec{R} in SC$. We have $N\sigma$ and $L\sigma$ are in SC

by i.h. Then $(N\ L)\sigma\vec{R} = N\sigma\ L\sigma\vec{R}$ is SN by definition.

– If $M = \lambda x.N$, then let $\vec{R} in SC$. We have $(\lambda x.N)\sigma\vec{R} = (\lambda x.N\sigma)\vec{R}$. By i.h. $N\sigma \cup \{x/x\} = N\sigma$ is SC, so $N\sigma$ and $\vec{R}$ are SN. Suppose $(\lambda x.N\sigma)\vec{R} \notin SN$. Then there is an infinite sequence of the form $(\lambda x.N\sigma)P\vec{Q} \rightarrow_\beta^* (\lambda x.N')P'\vec{Q'} \rightarrow_\beta\ N'\{x/P'\}\vec{Q'}$ where $\vec{R} = P\ \vec{Q}$. But $N\sigma\{x/P\}\vec{Q} \rightarrow_\beta^* N'\{x/P'\}\vec{Q'}$ and $N\sigma\{x/P\}\vec{Q}$ is SN by i.h.

▪

Second proof of the SN property (by Severi and van Raamsdonk)

1. Define SN inductively :
   - $M_1, \ldots, M_n \in SN$ implies $x\ M_1\ \ldots M_n \in SN$.
   - $M \in SN$ implies $\lambda x.M \in SN$.
   - $M\{x/N\}\vec{P} \in SN$ and $N \in SN$ implies $(\lambda x.M)\ N\ \vec{P} \in SN$.
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 (by R. David)

1. If $a, b, \vec{c}$ are SN and $(a\ b\ \vec{c})$ is not SN, then there is a' such that $a \to^* \lambda y.a'$ and $a'\{y/b\}\ \vec{c}$ is not SN.
2. If t^A and u^B are typed and SN, then $t\{x^B/u^B\}$ is SN.

   Proof. By induction on $\langle type(u), \mu(t), size(t)\rangle$.
   - Case $t = \lambda y.v$ is straightforward ($size(t)$ strictly decreases).
   - Case $t = y\ \vec{c}$ with $x \neq y$ is straightforward ($\mu(t)$ decreases and $size(t)$ strictly decreases.).
   - Case $t = x\ b\ \vec{c}$. By i.h. $b_1 = b\{x/u\}$ and $\vec{c_1} = \vec{c}\{x/u\}$ are SN. We want to show that $u\ b_1\ \vec{c_1}$ is SN. Suppose it is not. By previous point $u \to^* \lambda y.u'$ and $t' = u'\{y/b_1\}\vec{c_1}$ is not SN. But $type(b_1) < type(u)$ and u' is typed by subject

reduction so that $u'\{y/b_1\} \in SN$ by i.h. We rewrite $t' = (z\ \vec{c_1})\{z/u'\{y/b_1\}\}$. We have $type(u'\{y/b_1\}) = type(u') < type(u)$ and we conclude $t \in SN$ by i.h. Contradiction.

- Case $t = (\lambda z.b)\ c\ \vec{d}$. By i.h. $b_1 = b\{x/u\}$ and $c_1 = c\{x/u\}$ and $\vec{d_1} = \vec{d}\{x/u\}$ are SN. They are also typed. We want to show $t\{x/u\} = (\lambda z.b\{x/u\})\ c\{x/u\}\ \vec{d}\{x/u\}$ is SN. Suppose it is not. By previous point $b\{x/u\}\{z/c\{x/u\}\}\ \vec{d}\{x/u\}$ is not SN. This term is equal to $t'\{x/u\}$ where $t' = b\{z/c\}\ \vec{d}$. Since $\mu(t') < \mu(t)$ by i.h. the term is SN. Contradiction.

∎

3. If t is typable, then t is SN.

Proof. By induction on the structure of t. For the case $t = (u\ v)$ use the fact that $t = (z\ v)\{z/u\}$ and apply previous point 2. ∎

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`