1 λ-calculi

Definition 1.1 (λ-terms). *The language of λ-terms is inductively defined by:*

$$\begin{array}{rcl} M & ::= & V \mid MM \\ V & ::= & x \mid \lambda x.M \end{array}$$

where x belongs to an infinite enumerable set of variables. The language produced by the rule M is called Λ, while the sublanguage of values (rule V) is denoted by $\mathcal{V}$. We use the symbol $\equiv$ for syntactic equivalence.

Definition 1.2 (Free variables). *The set of* free variables *of a term M is denoted* $\mathrm{fv}(M)$*, and it's defined as follows:*

$$\begin{array}{rcl} \mathrm{fv}(x) & = & \{x\} \\ \mathrm{fv}(\lambda x.M) & = & \mathrm{fv}(M) \setminus \{x\} \\ \mathrm{fv}(MN) & = & \mathrm{fv}(M) \cup \mathrm{fv}(N) \end{array}$$

A term M is closed *iff* $\mathrm{fv}(M) = \emptyset$*, otherwise it's called* open*.*

Definition 1.3 (Substitution). *Let M, N and P be terms and x, y variables. $M[P/x]$ is called* substitution *and it's defined by:*

$$\begin{array}{rcl} y[P/x] & \equiv & \begin{cases} P & \textit{if } x \equiv y \\ y & \textit{otherwise} \end{cases} \\ (\lambda y.M)[P/x] & \equiv & \begin{cases} \lambda y.M & \textit{if } x \equiv y \\ \lambda y.(M[P/x]) & \textit{if } x \not\equiv y \textit{ and } y \notin FV(P) \\ \lambda z.(M[z/y][P/x]) & \textit{if } x \not\equiv y \textit{ and } y \in FV(P) \end{cases} \\ (MN)[P/x] & \equiv & (M[P/x])(N[P/x]) \end{array}$$

(where z is a fresh variable).

Definition 1.4 (Alpha equivalence). α-equivalence *is the smallest congruence relation $=_\alpha$ on lambda terms, such that for all terms M and all variables y that do not occur in M:*

$$\lambda x.M =_\alpha \lambda y.(M[y/x])$$

α-equivalent terms have identical interpretation and play identical roles in any application of λ-calculus. The notion of α-equivalence is what we usually mean when we refer to *equal terms*; in fact from now on we will write $=$ instead of $=_\alpha$.

Definition 1.5 (Small steps λ-calculus)**.** *Let $M, N \in \Lambda$, $V \in \mathcal{V}$. The* rewriting rules *for the* weak call-by-value λ-calculus *(weak cbv for short) are:*

$$(\lambda x.M)V \to_v M[V/x] \qquad (\beta^{\to} cbv)$$

$$\frac{M \to_v M'}{MN \to_v M'N}\ (\mathcal{L}^{\to} cbv) \qquad \frac{N \to_v N'}{VN \to_v VN'}\ (\mathcal{C}^{\to} cbv)$$

The rewriting rules for the call-by-name λ-calculus *(cbn for short) are:*

$$(\lambda x.M)N \to_n M[N/x] \qquad (\beta^{\to} cbn)$$

$$\frac{M \to_n M'}{MN \to_n M'N}\ (\mathcal{L}^{\to} cbn)$$

Let $\sigma \in \{v, n\}$ be a reduction strategy *and $c \geq 0$ a natural number. We write $M \xrightarrow{c}_\sigma N$ for:*

$$\begin{cases} M & \xrightarrow{0}_\sigma & M \\ M & \xrightarrow{a+1}_\sigma & N \quad \text{if } M \xrightarrow{a}_\sigma M' \text{ and } M' \to_\sigma N \end{cases}$$

furthermore, we write:

- *$M \to_\sigma^+ N$ if $M \xrightarrow{c}_\sigma N$ for some $c > 0$;*
- *$M \to_\sigma^* N$ if $M \xrightarrow{c}_\sigma N$ for $c \geq 0$ (in particular $M \longrightarrow_\sigma^* M$ for each $M \in \Lambda$).*

Definition 1.6 (Normal forms, stuck terms)**.** *The notion of reduction strategy comes together with the definition of* terms in normal form*: for each strategy σ, a term M is in normal form ($M \in \mathcal{N}_\sigma$) iff it doesn't exist a term M' such that $M \to_\sigma M'$.*

If a term is in normal form but not a value, it's said to be stuck*. A (necessarily open) term is stuck under a given strategy iff it is neither a value nor reducible by any of the reduction rules for that strategy. A quick inspection of the Definition 1.1 shows that such terms must be of the following form (where $M \in \Lambda, V \in \mathcal{V}$):*

$$\begin{array}{lrcl} \textit{(for call-by-value)} & S_v & ::= & xV \mid VS_v \mid S_vM \\ \textit{(for call-by-name)} & S_n & ::= & xM \mid S_nM \end{array}$$

Syntactically speaking, we can characterize the set of normal forms under a given strategy $\sigma \in \{v, n\}$, by taking S_v and S_n as the language generated respectively by the production rules above, and saying that:

$$\mathcal{N}_\sigma = \mathcal{V} \cup S_\sigma$$

2 CPS translation

Definition 2.1 (CPS translation)**.**

$$\begin{aligned}
[\![x]\!] &= x \\
[\![\lambda x.M]\!] &= \lambda k.k(\lambda x.[\![M]\!]) \\
[\![MN]\!] &= \lambda k.[\![M]\!](\lambda m.m[\![N]\!]k)
\end{aligned}$$

Proposition 2.2 (CPS-terms are values)**.**

$$M \in \Lambda \Rightarrow [\![M]\!] \in \mathcal{V}$$

Proof. Immediate by inspection of the rules of the CPS translation. □

Lemma 2.3 (CPS substitution)**.** *Let* $M, N \in \Lambda$*. Then:*

$$[\![M]\!][[\![N]\!]/x] = [\![M[N/x]]\!]$$

Proof. By structural induction on M:

- $M \equiv x$. Then $[\![x]\!][[\![N]\!]/x] = x[[\![N]\!]/x] = [\![N]\!] = [\![x[N/x]]\!]$
- $M \equiv y$. Then $[\![y]\!][[\![N]\!]/x] = y[[\![N]\!]/x] = y = [\![y]\!] = [\![y[N/x]]\!]$
- $M = \lambda y.P$. Then:

$$\begin{aligned}
[\![\lambda y.P]\!][[\![N]\!]/x] &= (\lambda k.k(\lambda y.[\![P]\!]))[[\![N]\!]/x] \\
&= \lambda k.k(\lambda y.[\![P]\!][[\![N]\!]/x]) \\
&\stackrel{(IH)}{=} \lambda k.k(\lambda y.[\![P[N/x]]\!]) \\
&= [\![\lambda y.P[N/x]]\!] \\
&= [\![(\lambda y.P)[N/x]]\!]
\end{aligned}$$

- $M = PQ$. Then:

$$\begin{aligned}
[\![PQ]\!][[\![N]\!]/x] &= (\lambda k.[\![P]\!](\lambda m.m[\![Q]\!]k))[[\![N]\!]/x] \\
&= \lambda k.[\![P]\!][[\![N]\!]/x](\lambda m.m[\![Q]\!][[\![N]\!]/x]k) \\
&\stackrel{(IH)}{=} \lambda k.[\![P[N/x]]\!](\lambda m.m[\![Q[N/x]]\!]k) \\
&= [\![P[N/x]Q[N/x]]\!] \\
&= [\![(PQ)[N/x]]\!]
\end{aligned}$$

□

Definition 2.4 (Functional depth)**.**

$$\begin{array}{rcl} \delta(x) & = & 0 \\ \delta(\lambda x.M) & = & 0 \\ \delta(MN) & = & 1 + \delta(M) \end{array}$$

Lemma 2.5. *Let $M, M' \in \Lambda$. Then:*

$$M \to_n M' \quad \textit{implies} \quad \delta(M') \geq \delta(M) - 1$$

Proof. By induction on the reduction $M \to_n M'$:

- $(\lambda x.M)N \to_n M[N/x]$. We have to proof:

$$\delta(M[N/x]) \geq \delta((\lambda x.M)N) - 1 = 0$$

  but that's obvious, because for each $M \in \Lambda$ we have $\delta(M) \geq 0$.

- $\dfrac{M \to_n M'}{MN \to_n M'N}$. Then, by inductive hypothesis:

$$\begin{array}{rcl} \delta(M') & \geq & \delta(M) - 1 \\ 1 + \delta(M') & \geq & \delta(M) \\ \delta(M'N) & \geq & \delta(MN) - 1 \end{array}$$

□

Lemma 2.6. *Let $M, M' \in \Lambda$: if $M \to_n M'$ then exists $M_t \in \Lambda$ such that, for all continuations $K \in \mathcal{V}$ the following holds:*

$$[\![M]\!]K \overset{\delta(M)+3}{\longrightarrow_v} M_t \quad \textit{and} \quad [\![M']\!]K \overset{\delta(M)-1}{\longrightarrow_v} M_t$$

Proof. By induction on the reduction $M \to_n M'$:

- $(\lambda x.M)N \to_n M[N/x]$. Then:

$$\begin{array}{rcl} [\![(\lambda x.M)N]\!]K & \to_v & [\![(\lambda x.M)]\!](\lambda m.m[\![N]\!]K) \\ & \to_v & (\lambda m.m[\![N]\!]K)(\lambda x.[\![M]\!]) \\ & \to_v & (\lambda x.[\![M]\!])[\![N]\!]K \\ & \to_v & [\![M]\!][[\![N]\!]/x]K \end{array}$$

  that is equal to $[\![M[N/x]]\!]K$ by Lemma 2.3. In this case we have $\delta((\lambda x.M)N) = 1$, and in fact we used 4 steps in order to obtain $M_t = [\![M[N/x]]\!]K$ from $[\![(\lambda x.M)N]\!]K$, and zero steps from $[\![M[N/x]]\!]K$.

- $\dfrac{M \to_n M'}{MN \to_n M'N}$. Then:

$$\begin{tikzcd}
{[\![MN]\!]K} \arrow[r, "\to_v", phantom] & {[\![M]\!](\lambda m.m[\![N]\!]K)} \arrow[dr, "\delta(M)+3"', "v"] & \\
& & M_t \quad \text{(by I.H.)} \\
{[\![M'N]\!]K} \arrow[r, "\to_v", phantom] & {[\![M']\!](\lambda m.m[\![N]\!]K)} \arrow[ur, "\delta(M)-1", "v"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
{[\![MN]\!]K} \arrow[r, "v"'] & {[\![M]\!](\lambda m.m[\![N]\!]K)} \arrow[dr, "\delta(M)+3", "v"'] & \\
& & M_t \quad \text{(by I.H.)} \\
{[\![MN]\!]K} \arrow[r, "v"'] & {[\![M']\!](\lambda m.m[\![N]\!]K)} \arrow[ur, "\delta(M)-1", "v"'] &
\end{tikzcd}$$

We can conclude, observing that $1 + \delta(M) + 3 = \delta(MN) + 3$ and $1 + \delta(M) - 1 = \delta(MN) - 1$.

$\square$

References

[1] Gordon Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.