Komponenten- und Service-orientierte Softwarekonstruktion

Lecture 2: λ-Calculus

Jakob Rehof
LS XIV – Software Engineering

TU Dortmund
Sommersemester 2015

SS 2015

λ-terms

Let $\mathcal{V}$ denote a denumerable set of λ-*variables*, $v_1, v_2, v_3, \ldots$.

Let $x, y, z, \ldots$ denote a denumerable set of *metavariables* ranging over $\mathcal{V}$.

Definition 1

The set Λ of λ-*terms*, ranged over by $M, N, P, Q, \ldots$, is defined inductively by:

- $x \in \mathcal{V} \Rightarrow x \in \Lambda$ (variables)
- $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$ (application)
- $M \in \Lambda, x \in \mathcal{V} \Rightarrow (\lambda x M) \in \Lambda$ (abstraction)

λ-terms

Notation 1

We may abbreviate $(((FM_1)M_2)\cdots M_n)$ by

$$FM_1M_2\cdots M_n$$

for $n \geq 0$.

We may abbreviate $(\lambda x_1(\lambda x_2(\cdots(\lambda x_n M)\cdots)))$ by

$$\lambda x_1.\lambda x_2.\cdots\lambda x_n.M$$

for $n \geq 0$.

And further, $\lambda x_1.\lambda x_2.\cdots\lambda x_n.M$ by $\lambda x_1 x_2.\cdots x_n.M$.

Definition 2 (Free and bound variables)

The set of *free variables* of M, denoted $\mathsf{FV}(M)$, is defined by induction on M:

$$\begin{array}{lcl} \mathsf{FV}(x) & = & \{x\} \\ \mathsf{FV}(MN) & = & \mathsf{FV}(M) \cup \mathsf{FV}(N) \\ \mathsf{FV}(\lambda x.M) & = & \mathsf{FV}(M) \setminus \{x\} \end{array}$$

A variable $x \in \mathsf{FV}(M)$ is called *bound* in $\lambda x.M$ (by that λ). A term $M \in \Lambda$ is called *closed*, if $\mathsf{FV}(M) = \emptyset$.

Definition 3 (Syntactic identity)

We write $M \equiv N$ to denote that M and N are the same term under renaming of bound variables (α-*conversion*). For example,

$$\begin{array}{lcll} (\lambda x.x)z & \equiv & (\lambda x.x)z & \\ (\lambda x.x)z & \equiv & (\lambda y.y)z & \\ (\lambda x.x)x & \equiv & (\lambda y.y)x & \\ (\lambda x.x)z & \not\equiv & (\lambda x.y)z & \text{(provided } x \not\equiv y\text{)} \end{array}$$

Note that $x \equiv y$ is possible, unless we have stipulated that $x \not\equiv y$.

Definition 4 (Substitution)

Let $M[x := N]$ denote the result of substituting N for the free occurrences of x in M, defined by induction on M:

$$
\begin{array}{lcl}
x[x := N] & \equiv & N \\
y[x := N] & \equiv & y, \quad \text{if } x \not\equiv y \\
(PQ)[x := N] & \equiv & (P[x := N]Q[x := N]) \\
(\lambda y.P)[x := N] & \equiv & \lambda y.P[x := N], \quad \text{if } x \not\equiv y \\
(\lambda x.P)[x := N] & \equiv & \lambda x.P
\end{array}
$$

Lemma 5 (Substitution lemma)

Assume $x \not\equiv y$ *and* $x \notin \mathsf{FV}(P)$. *Then*

$$M[x := N][y := P] \equiv M[y := P][x := N[y := P]]$$

Exercise 1

Prove Lemma 5 by induction on M.

Notion of reduction

Definition 6

A *notion of reduction* on a set $\mathbf{D}$ is a binary relation $\rhd \subseteq \mathbf{D} \times \mathbf{D}$.

Definition 7 (Notion of β-reduction)

The notion of β-*reduction* is the relation $\rhd_\beta \subseteq \Lambda \times \Lambda$ defined by

$$(\lambda x.M)N \ \rhd_\beta \ M[x := N]$$

for all $M, N \in \Lambda$. An expression of the form $(\lambda x.M)N$ is called a *redex*.

Contexts

Contexts C over Λ are expressions of the form

$$C ::= [] \mid (CM) \mid (MC) \mid \lambda x.C$$

We think of $[]$ as a hole and of C as a term with a hole in it.

If C is a context and M a term then $C[M]$ is the term which arises from exchanging $[]$ with M ("hole-filling").

For example, if $C \equiv \lambda x.\lambda y.y[]$, then

$$C[\lambda z.(xz)] \equiv \lambda x.\lambda y.y(\lambda z.(xz))$$

Note that variable capture can happen through hole-filling.

β-reduction

Definition 8

A notion of reduction $\rhd \subseteq \Lambda \times \Lambda$ satisfying

$$\forall C.\ M \ \rhd \ N \Rightarrow C[M] \ \rhd \ C[N]$$

is called a *reduction relation*. The smallest reduction relation containing $\rhd$ is called the *reduction relation induced by* $\rhd$ (sometimes also called the compatible closure of $\rhd$).

Definition 9 (β-reduction)

The relation $\to_\beta$ (β-*reduction*) is the reduction relation induced by $\rhd_\beta$.

If $M \to_\beta N$, then N is called a *reduct* of M.

A *normal form* is a term with no reduct.

β-reduction

Definition 10 (Multi-step β-reduction)

The relation $\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\rightarrow_\beta$.

Definition 11 (β-conversion)

The relation $=_\beta$ (β-*conversion*) is the reflexive symmetric transitive closure of $\rightarrow_\beta$.

If $M =_\beta N$, we say that M and N are β-equivalent.

λ-calculus combinators

A λ-calculus *combinator* is a closed λ-term.

Some interesting combinators are:

$\mathbf{I} \equiv \lambda x.x$, $\mathbf{K} \equiv \lambda xy.x$, $\mathbf{K}^* \equiv \lambda xy.y$, $\mathbf{S} \equiv \lambda xyz.(xz)(yz)$

$\omega \equiv \lambda x.xx$, $\Omega \equiv \omega\omega$

$\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, $\Theta \equiv (\lambda x.\lambda y.(y(xxy)))(\lambda x.\lambda y.(y(xxy)))$

Exercise 2

Show that, for all terms F, we have $\mathbf{Y}F =_\beta F(\mathbf{Y}F)$ and $\Theta F \twoheadrightarrow_\beta F(\Theta F)$.

Confluence

Theorem 12 (Church-Rosser, confluence)

For every $M, P_1, P_2 \in \Lambda$, if $M \twoheadrightarrow_\beta P_1$ and $M \twoheadrightarrow_\beta P_2$, then there exists $Q \in \Lambda$ such that $P_1 \twoheadrightarrow_\beta Q$ and $P_2 \twoheadrightarrow_\beta Q$.

Corollary 13 (Uniqueness of normal forms)

If N_1 and N_2 are normal forms such that $M \twoheadrightarrow_\beta N_1$ and $M \twoheadrightarrow_\beta N_2$, then $N_1 \equiv N_2$. If $P =_\beta Q$, then P and Q have the same normal form if any.

Exercise 3

Prove Corollary 13, by using Theorem 12.

Church numerals

For $P, M \in \Lambda$, define $P^n(M)$ by induction on $n \geq 0$:

- $P^0(M) \equiv M$
- $P^{n+1}(M) \equiv P(P^n(M))$

The *Church numerals* $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \ldots$ are defined by $\mathbf{c}_n \equiv \lambda f.\lambda x.f^n(x)$.

Define $\mathbf{A}_+ \equiv \lambda xypq.xp(ypq)$, $\mathbf{A}_\times \equiv \lambda xyz.x(yz)$, $\mathbf{A}_e \equiv \lambda xy.yx$. Then one can prove:

$\mathbf{A}_+\mathbf{c}_n\mathbf{c}_m =_\beta \mathbf{c}_{n+m}$,

$\mathbf{A}_\times\mathbf{c}_n\mathbf{c}_m =_\beta \mathbf{c}_{n\times m}$,

$\mathbf{A}_e\mathbf{c}_n\mathbf{c}_m =_\beta \mathbf{c}_{n^m}$ $(m > 0)$.

Turing completeness

A numerical function $f : \mathbb{N}^k \to \mathbb{N}$ is called λ-*definable* if there exists a combinator P_f such that

$$P_f \mathbf{c}_{n_1} \dots \mathbf{c}_{n_k} =_\beta \mathbf{c}_{f(n_1,\dots,n_k)}$$

Theorem 14 (Kleene)

Every recursive function is λ-definable.