Typed λ-calculus: Substitution and Equations

P. B. Levy
adapted for G53POP by T. Altenkirch
Universities of Birmingham and Nottingham

1 Renaming and Substitution

Suppose we have a term \( \Gamma \vdash M : B \), and we want to turn it into a term in context \( \Delta \), by replacing the identifiers. For example, we’re given the term

\[
x : \text{int}, y : \text{bool}, z : \text{int} \vdash z + \text{case } y \text{ of } \{ \text{true } \rightarrow x + z | \text{false } \rightarrow x + 1 \} : \text{int}
\]

and we want to change it to something in the context \( u : \text{bool}, x : \text{int}, y : \text{bool} \).

1.1 Replacing Identifiers With Identifiers

One way is to replace identifiers in \( \Gamma \) with identifiers in \( \Delta \). A \textit{renaming} from \( \Gamma \) to \( \Delta \) (beware the direction here) is a function \( \theta \) taking each identifier \( x : A \) in \( \Gamma \) to an identifier \( \theta(x) : A \) in \( \Delta \).

For example, using the above \( \Gamma \) and \( \Delta \), one renaming from \( \Gamma \) to \( \Delta \) is

\[
x \mapsto x \\
y \mapsto u \\
z \mapsto x
\]

We write \( \theta^* M \) for the result of changing all the free identifiers in \( M \) according to \( \theta \). In the above example, we obtain

\[
u : \text{bool}, x : \text{int}, y : \text{bool} \vdash x + \text{case } u \text{ of } \{ \text{true } \rightarrow x + x | \text{false } \rightarrow x + 1 \} : \text{int}
\]
Exercise 1. Apply to the term
\[ x : int \to int, y : int \vdash \text{let } w = 5 \text{ in } (x y) + (x w) : int \]
the renaming
\[
\begin{align*}
x & \mapsto y \\
y & \mapsto w
\end{align*}
\]
to obtain a term in context
\[ w : int, y : int \to int, z : int \]

1.2 Replacing Identifiers With Terms

The second example is called substitution, where we replace each identifier in \( \Gamma \) with a term in context \( \Delta \). A substitution from \( \Gamma \) to \( \Delta \) is a function \( k \) taking each identifier \( x : A \) in \( \Gamma \) to a term \( \Delta \vdash k(x) : A \).

For example, using the above \( \Gamma \) and \( \Delta \), a substitution from \( \Gamma \) to \( \Delta \) is
\[
\begin{align*}
x & \mapsto 3 + x \\
y & \mapsto u \\
z & \mapsto \text{case } y \text{ of } \{ \text{true } \to x + 2 \mid \text{false } \to x \}
\end{align*}
\]
We write \( k^* M \) for the result of replacing all the free identifiers in \( M \) according to \( k \) (avoiding capture, of course). In the above example, we obtain
\[
\begin{align*}
u : \text{bool}, x : \text{int}, y : \text{bool} \vdash \\
\text{case } y \text{ of } \{ \text{true } \to x + 2 \mid \text{false } \to x \} + \\
\text{case } u \text{ of } \{ \text{true } \to (3 + x) + \text{case } y \text{ of } \{ \text{true } \to x + 2 \mid \text{false } \to x \} \\
\mid \text{false } \to (3 + x) + 1 \} : \text{int}
\end{align*}
\]

Exercise 2. Apply to the term
\[ x : int \to int, y : int \vdash \text{let } w = 5 \text{ in } (x y) + (x w) : int \]
the substitution
\[
\begin{align*}
x & \mapsto y \\
y & \mapsto w + 1
\end{align*}
\]
to obtain a term in context
\[ w : int, y : int \to int, z : int \]
1.3 Substitution Uses Renaming

It is clear that renaming is a special case of substitution. So why is it important to consider both? The reason appears when we wish to define \( k^*M \) by induction on \( M \). Some of the inductive clauses are easy:

\[
\begin{align*}
  k^*3 &= 3 \\
  k^*(M + N) &= k^*M + k^*N \\
  k^*x &= k(x)
\end{align*}
\]

But what about substituting into a \texttt{let} expression? Let’s first remember the typing rule for \texttt{let}:

\[
\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B \\
\Gamma \vdash \text{let } x = M \text{ in } N : B
\]

(I’m going to assume that \( x \) doesn’t appear in \( \Gamma \) or \( \Delta \). Otherwise, you can \( \alpha \)-convert it to something else.)

We want to define

\[
k^*(\text{let } x = M \text{ in } N) = \text{let } x = k^*M \text{ in } (k, x : A)^*N
\]

where the substitution \( \Gamma, x : A \xrightarrow{k,x:A} \Delta, x : A \) is... what? Remember that it has to map each identifier in \( \Gamma, x : A \) to a term (of the same type) in context \( \Delta, x : A \). Clearly it maps \( x \) to \( x \). And it maps \((y : B) \in \Gamma\) to \( k(y)\)—which is in context \( \Delta \)—renamed along the renaming from \( \Delta \) to \( \Delta, x : A \).

So we have to define renaming before we can define \( k, x : A \), and we have to define \( k, x : A \) before we can define substitution.

How do we define renaming inductively? Again, some of the inductive clauses are easy:

\[
\begin{align*}
  \theta^*3 &= 3 \\
  \theta^*(M + N) &= \theta^*M + \theta^*N \\
  \theta^*x &= \theta(x)
\end{align*}
\]

For \texttt{let}, we want to define

\[
\theta^*(\text{let } M \text{ be } x \text{ in } N) = \text{let } x = \theta^*M \text{ in } (\theta, x : A)^*N
\]
where the renaming morphism $\Gamma, x : A \xrightarrow{\theta, x : A} \Delta, x : A$ maps $x$ to $x$, and otherwise is the same as $\theta$.

In summary, the definition of substitution goes in 4 stages:

- define $\theta, x : A$
- define renaming by induction
- define $k, x : A$
- define substitution by induction.

A consequence of this is that if you want to prove a theorem about substitution, you’ll first have to prove it for renaming.

**Proposition 1.** 1. Contexts and substitutions form a category—
composition is defined by substitution. This means

$$k; \text{id} = k$$
$$\text{id}; k = k$$
$$(k; l); m = k; (l; m)$$

Renamings form a subcategory, i.e. every renaming is a substitution and renamings have the same set of laws.

2. $(k; l)^* M$ is the same as $k^* l^* M$, and $\text{id}^* M$ is the same as $M$.

## 2 Evaluation Through $\beta$-reduction

Intuitively, a $\beta$-reduction means simplification. I’ll write $M \rightsquigarrow N$ to mean that $M$ can be simplified to $N$. For example, there are $\beta$-reduction rules for all the arithmetic operations:

$$m + n \rightsquigarrow m + n$$
$$m \times n \rightsquigarrow m \times n$$
$$m > n \rightsquigarrow \text{true} \text{ if } m > n$$
$$m > n \rightsquigarrow \text{false} \text{ if } m \leq n$$

There is a $\beta$-reduction rule for local definitions:

$$\text{let } x = M \text{ in } N \rightsquigarrow N[M/x]$$
But the most interesting are the $\beta$-reductions for all the types. The rough idea is: if you use an introduction rule and then, immediately, use an elimination rule, then they can be simplified.

For the boolean type, the $\beta$-reduction rule is

\[
\text{case true of } \{ \text{true } \rightarrow N \mid \text{false } \rightarrow N' \} \rightsquigarrow N
\]

\[
\text{case false of } \{ \text{true } \rightarrow N \mid \text{false } \rightarrow N' \} \rightsquigarrow N'
\]

For the type $A \times B$, if we use projections the $\beta$-reduction rule is

\[
\text{fst } (M, M') \rightsquigarrow M
\]

\[
\text{snd } (M, M') \rightsquigarrow M'
\]

If we use pattern-matching, the $\beta$-reduction rule is

\[
\text{case } (M, M') \text{ of } (\text{x}, \text{y}) \rightarrow N \rightsquigarrow N[M/x, M'/y]
\]

For the type $A + B$, the $\beta$-reduction rule is

\[
\text{case } \text{left, } M \text{ of } \{ (\text{left, } x) \text{ in } N, (\text{right, } y) \text{ in } N' \} \rightsquigarrow N[M/x]
\]

\[
\text{case } \text{right, } M \text{ of } \{ (\text{left, } x) \text{ in } N, (\text{right, } y) \text{ in } N' \} \rightsquigarrow N'[M/y]
\]

For the type $A \rightarrow B$, the $\beta$-reduction rule is

\[
(\lambda x. M) N \rightsquigarrow M[N/x]
\]

A term which is the left-hand-side of a $\beta$-reduction is called a $\beta$-redex.

You can simplify any term $M$ by picking a subterm that’s a $\beta$-redex, and reduce it. Do this again and again until you get a $\beta$-normal term, i.e. one that doesn’t contain any $\beta$-redex. It can be shown that this process has to terminate (the strong normalization theorem).

**Proposition 2.** A closed term $M$ that is $\beta$-normal must have an introduction rule at the root. (Remember that we consider $\text{n}$ to be an introduction rule, but not $+ \times >$.) Hence, if $M$ has type $\text{int}$, then it must be $\text{n}$ for some $n \in \mathbb{Z}$. 
We prove the first part by induction on $M$.

Exercise 3. All the sums that we did can be turned into expressions and evaluated using $\beta$-reduction. Try:

1. let $x = (5, (2, \text{true}))$ in $\text{fst } x + \text{fst (case } x \text{ of } (y, z) \rightarrow z)$

2. \[
\text{case (case (3 < 7) of } \{ \text{true } \rightarrow (\#\text{right}, 8 + 1) \mid \text{false } \rightarrow (\#\text{left}, 2) \} \text{ of }
\{ (\#\text{left}, u) \rightarrow u + 8 \mid (\#\text{right}, u) \rightarrow u + 3 \})
\]

3. $(\lambda f : \text{int } \rightarrow \text{int.} \lambda x : \text{int.} f(x)(\lambda x : \text{int.} x + 3)2$

3 $\eta$-expansion

The $\eta$-expansion laws express the idea that

- everything of type $\text{bool}$ is $\text{true}$ or $\text{false}$
- everything of type $A \times B$ is a pair $(x, y)$
- everything of type $A + B$ is a pair $(\#\text{left}, x)$ or a pair $(\#\text{right}, x)$
- everything of type $A \rightarrow B$ is a function.

They are given by first applying an elimination, then an introduction (the opposite of $\beta$-reduction).

Let’s begin with the type $\text{bool}$. If we have a term $\Gamma, z : \text{bool} \vdash N : B$, it can be $\eta$-expanded to

\[
\text{case } z \text{ of } \{ \text{true } \rightarrow N[\text{true}/z] \mid \text{false } \rightarrow N[\text{false}/z] \}
\]

The reason this ought to be true is that, whatever we define the identifiers in $\Gamma$ to be, $z$ will be either $\text{true}$ or $\text{false}$. Either way, both sides should be the same.

What about $A \times B$? If we’re using projections, then any $\Gamma \vdash M : A \times B$ can be $\eta$-expanded to $(\text{fst } M, \text{snd } M)$.

And if we’re using pattern-match, suppose $\Gamma, z : A \times B \vdash N : C$. Then $N$ can be expanded into

\[
\text{case } z \text{ of } (x, y)N[(x, y)/z]
\]

(I’m supposing the $x$ and $y$ we use here don’t appear in $\Gamma, z : A \times B$. )
For $A + B$, it’s similar. Suppose $\Gamma, z : A + B \vdash N : C$. Then $N$ can be expanded into
\[
\text{case } z \text{ of } \{(\#\text{left}, x) \rightarrow N[(\#\text{left}, x)/z] | (\#\text{right}, y) \rightarrow N[(\#\text{right}, y)/z]\}
\]
(Again, I’m supposing the $x$ and $y$ don’t appear in $\Gamma, z : A + B$.)

And finally, $A \rightarrow B$. Any term $\Gamma \vdash M : A \rightarrow B$ can be expanded as $\lambda x.(Mx)$.

(Again, I’m supposing the $x$ doesn’t appear in $\Gamma$.)

\textbf{Exercise 4.} Take the term
\[
f : (\text{int} + \text{bool}) \rightarrow (\text{int} + \text{bool}) \vdash f : (\text{int} + \text{bool}) \rightarrow (\text{int} + \text{bool})
\]
Apply an $\eta$-expansion for $\rightarrow$, then for $+$, then for $\text{bool}$.

\section{Equality}

$\lambda$-calculus isn’t just a set of terms; it comes with an equational theory. If $\Gamma \vdash M : B$ and $\Gamma \vdash N : B$, we write $\Gamma \vdash M = N : B$ to express the intuitive idea that, no matter what we define the identifiers in $\Gamma$ to be, $M$ and $N$ have the same “meaning” (even though they’re different expressions).

First of all we need rules to say that this is an equivalence relation:
\[
\begin{align*}
\Gamma \vdash M : B & \quad \Gamma \vdash M = M : B \\
\Gamma \vdash M = M : B & \quad \Gamma \vdash M = N : B \\
\Gamma \vdash M = N : B & \quad \Gamma \vdash N = M : B \\
\Gamma \vdash M = N : B & \quad \Gamma \vdash N = P : B \\
\Gamma \vdash M = P : B &
\end{align*}
\]

Secondly, we need rules to say that this is \textit{compatible}—preserved by every construct:
\[
\Gamma \vdash M = M' : A \quad \Gamma, x : A \vdash N = N' : B \\
\Gamma \vdash \text{let } x = M \text{ in } N = \text{let } x = M' \text{ in } N' : B
\]
and so forth. A compatible equivalence relation is often called a \textit{congruence}.
Thirdly, each of the $\beta$-reductions that we’ve seen is an axiom of this theory.

$$\Gamma \vdash N : B \quad \Gamma \vdash N' : B$$

$$\Gamma \vdash \text{case true of } \{ \text{true} \to N \mid \text{false} \to N' \} = N : B$$

$$\Gamma, x : A \vdash M : B \quad \Gamma \vdash N : A$$

$$\Gamma \vdash (\lambda x.M) N = M[N/x] : B$$

Fourthly, each of the $\eta$-expansions is an axiom of the theory, e.g.

$$\Gamma \vdash M : A \to B$$

$$\Gamma \vdash M = \lambda x.(Mx) : A \to B$$

But in the case of the $\eta$-expansions involving pattern-matching, we need to generalize them slightly. The reason is that we want to prove Proposition 3. If $\Gamma \vdash M = N : B$ and $\Gamma \xrightarrow{k} \Delta$ is a substitution, then $\Delta \vdash k^* M = k^* N : B$.

Consequently, the $\eta$-law for $\text{bool}$ looks like this:

$$\Gamma \vdash M : \text{bool} \quad \Gamma, z : \text{bool} \vdash N : C$$

$$\Gamma \vdash N[M/z] = \text{case } M \text{ of } \{ \text{true} \to N[true/z] \mid \text{false} \to N[false/z] \} : C$$

and similarly for the other pattern-matching laws. We can then prove Prop. 3, first for renamings, then for substitution.

5 Exercises

1. Suppose that $\Gamma \vdash M : \text{bool}$ and $\Gamma \vdash N_0, N_1, N_2, N_3 : C$. Show that

$$\Gamma \vdash \text{case } M \text{ of } \{$$

true $\to \text{case } M \text{ of } \{ \text{true}.N_0 \mid \text{false}.N_1 \}$,

false $\to \text{case } M \text{ of } \{ \text{true} \to N_2 \mid \text{false} \to N_3 \}$

$$\} = \text{case } M \text{ of } \{ \text{true} \to N_0 \mid \text{false} \to N_3 \} : C$$
2. Show that \((\lambda\left, -\right)\) is injective, i.e. if \(\Gamma \vdash M, M' : A\) and \(\Gamma \vdash (\lambda\left, M\right) = (\lambda\left, M'\right) : A + B\) then \(\Gamma \vdash M = M' : A\).

3. Write down the \(\eta\)-law for the 0 type.

4. Given a term \(\Gamma, x : A \vdash M : 0\), show that it is an “isomorphism” in the sense that there is a term \(\Gamma, y : 0 \vdash N : A\) satisfying

\[
\Gamma, y : 0 \vdash M[N/x] = y : 0 \\
\Gamma, x : A \vdash N[M/x] = x : A
\]

5. Give \(\beta\) and \(\eta\) laws for \(\alpha(A, B, C, D, E)\) and for \(\beta(A, B, C, D, E, F, G)\).
(See yesterday’s exercises for a description of these types.)