

Typed λ -calculus: Substitution and Equations

P. B. Levy

University of Birmingham

1 Renaming and Substitution

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

$$x : \text{int}, y : \text{bool}, z : \text{int} \vdash z + \text{match } y \text{ as } \{\text{true. } x+z, \text{false. } x+1\} : \text{int}$$

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 Γ with *identifiers* in Δ . A *renaming* from Γ to Δ (beware the direction here) is a function θ taking each identifier $x : A$ in Γ to an identifier $\theta(x) : A$ in Δ .

For example, using the above Γ and Δ , one renaming from Γ to Δ is

$$\begin{aligned} x &\mapsto x \\ y &\mapsto u \\ z &\mapsto x \end{aligned}$$

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

$$u : \text{bool}, x : \text{int}, y : \text{bool} \vdash x + \text{match } u \text{ as } \{\text{true. } x+x, \text{false. } x+1\} : \text{int}$$

Exercise 1. Apply to the term

$$x : \text{int} \rightarrow \text{int}, y : \text{int} \vdash \text{let } 5 \text{ be w. } (xy) + (xw) : \text{int}$$

the renaming

$$\begin{aligned}x &\mapsto y \\ y &\mapsto w\end{aligned}$$

to obtain a term in context

$$w : \text{int}, y : \text{int} \rightarrow \text{int}, z : \text{int}$$

1.2 Replacing Identifiers With Terms

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

For example, using the above Γ and Δ , a substitution from Γ to Δ is

$$\begin{aligned}x &\mapsto 3 + x \\ y &\mapsto u \\ z &\mapsto \text{match } y \text{ as } \{\text{true. } x + 2, \text{false. } x\}\end{aligned}$$

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

$$\begin{aligned}u : \text{bool}, x : \text{int}, y : \text{bool} &\vdash \\ \text{match } y \text{ as } \{\text{true. } x + 2, \text{false. } x\} &+ \\ \text{match } u \text{ as } \{\text{true. } (3 + x) + \text{match } y \text{ as } \{\text{true. } x + 2, \text{false. } x\}, & \\ \text{false. } (3 + x) + 1\} : \text{int} &\end{aligned}$$

Exercise 2. Apply to the term

$$x : \text{int} \rightarrow \text{int}, y : \text{int} \vdash \text{let } 5 \text{ be } w. (xy) + (xw) : \text{int}$$

the substitution

$$\begin{aligned}x &\mapsto y \\ y &\mapsto w + 1\end{aligned}$$

to obtain a term in context

$$w : \text{int}, y : \text{int} \rightarrow \text{int}, z : \text{int}$$

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:

$$\begin{aligned} k^*3 &= 3 \\ k^*(M + N) &= k^*M + k^*N \\ k^*\mathbf{x} &= k(\mathbf{x}) \end{aligned}$$

But what about substituting into a `let` expression? Let's first remember the typing rule for `let` :

$$\frac{\Gamma \vdash M : A \quad \Gamma, \mathbf{x} : A \vdash N : B}{\Gamma \vdash \text{let } M \text{ be } \mathbf{x}. N : B}$$

(I'm going to assume that \mathbf{x} doesn't appear in Γ or Δ . Otherwise, you can α -convert it to something else.)

We want to define

$$k^*(\text{let } M \text{ be } \mathbf{x}. N) = \text{let } k^*M \text{ be } \mathbf{x}. (k, \mathbf{x} : A)^*N$$

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

So we have to define renaming before we can define $k, \mathbf{x} : A$, and we have to define $k, \mathbf{x} : A$ before we can define substitution.

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

$$\begin{aligned} \theta^*3 &= 3 \\ \theta^*(M + N) &= \theta^*M + \theta^*N \\ \theta^*\mathbf{x} &= \theta(\mathbf{x}) \end{aligned}$$

For `let` , we want to define

$$\theta^*(\text{let } M \text{ be } \mathbf{x}. N) = \text{let } \theta^*M \text{ be } \mathbf{x}. (\theta, \mathbf{x} : A)^*N$$

where the renaming morphism $\Gamma, \mathbf{x} : A \xrightarrow{\theta, \mathbf{x} : A} \Delta, \mathbf{x} : A$ maps \mathbf{x} to \mathbf{x} , and otherwise is the same as θ .

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

- define $\theta, \mathbf{x} : A$
- define renaming by induction
- define $k, \mathbf{x} : A$
- 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. Renamings form a subcategory.*

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

2 Evaluation Through β -reduction

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

$$\begin{aligned} \underline{m} + \underline{n} &\rightsquigarrow \underline{m + n} \\ \underline{m} \times \underline{n} &\rightsquigarrow \underline{m \times n} \\ \underline{m} > \underline{n} &\rightsquigarrow \text{true if } m > n \\ \underline{m} > \underline{n} &\rightsquigarrow \text{false if } m \leq n \end{aligned}$$

There is a β -reduction rule for local definitions:

$$\text{let } M \text{ be } \mathbf{x}. N \rightsquigarrow N[M/\mathbf{x}]$$

But the most interesting are the β -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 β -reduction rule is

$$\begin{aligned} \text{match true as } \{\text{true}.N, \text{false}.N'\} &\rightsquigarrow N \\ \text{match false as } \{\text{true}.N, \text{false}.N'\} &\rightsquigarrow N' \end{aligned}$$

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

$$\begin{aligned}\pi \langle M, M' \rangle &\rightsquigarrow M \\ \pi' \langle M, M' \rangle &\rightsquigarrow M'\end{aligned}$$

If we use pattern-matching, the β -reduction rule is

$$\text{match } \langle M, M' \rangle \text{ as } \langle \mathbf{x}, \mathbf{y} \rangle. N \rightsquigarrow N[M/\mathbf{x}, M'/\mathbf{y}]$$

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

$$\begin{aligned}\text{match inl } M \text{ as } \{\text{inl } \mathbf{x}. N, \text{inr } \mathbf{y}. N'\} &\rightsquigarrow N[M/\mathbf{x}] \\ \text{match inr } M \text{ as } \{\text{inl } \mathbf{x}. N, \text{inr } \mathbf{y}. N'\} &\rightsquigarrow N'[M/\mathbf{y}]\end{aligned}$$

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

$$(\lambda \mathbf{x}. M)N \rightsquigarrow M[N/\mathbf{x}]$$

A term which is the left-hand-side of a β -reduction is called a β -redex.

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

Proposition 2. *A closed term M that is β -normal must have an introduction rule at the root. (Remember that we consider \underline{n} to be an introduction rule, but not $+\times >.$) Hence, if M has type int , then it must be \underline{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 β -reduction. Try:

1. $\text{let } \langle 5, \langle 2, \text{true} \rangle \rangle \text{ be } \mathbf{x}. \pi \mathbf{x} + \pi(\text{match } \mathbf{x} \text{ as } \langle \mathbf{y}, \mathbf{z} \rangle. \mathbf{z})$
2. $\text{match } (\text{match } (3 < 7) \text{ as } \{\text{true}. \text{inr } 8 + 1, \text{false}. \text{inl } 2\}) \text{ as } \{\text{inl } \mathbf{u}. \mathbf{u} + 8, \text{inr } \mathbf{u}. \mathbf{u} + 3\}$
3. $((\lambda \mathbf{f}_{\text{int} \rightarrow \text{int}}. \lambda \mathbf{x}_{\text{int}}. (\mathbf{f}(\mathbf{f}\mathbf{x}))) \lambda \mathbf{x}_{\text{int}}. (\mathbf{x} + 3))2$

3 η -expansion

The η -expansion laws express the idea that

- everything of type `bool` is `true` or `false`
- everything of type $A \times B$ is a pair $\langle x, y \rangle$
- everything of type $A + B$ is a pair `inl` x or a pair `inr` x
- everything of type $A \rightarrow B$ is a function.

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

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

$$\text{match } z \text{ as } \{\text{true}. N[\text{true}/z], \text{false}. N[\text{false}/z]\}$$

The reason this ought to be true is that, whatever we define the identifiers in Γ to be, z will be either `true` or `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 η -expanded to $\langle \pi M, \pi' M \rangle$.

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

$$\text{match } z \text{ as } \langle x, y \rangle N[\langle x, y \rangle/z]$$

(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

$$\text{match } z \text{ as } \{\text{inl } x.N[\text{inl } x/z], \text{inr } y.N[\text{inr } y/z]\}$$

(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 Γ .)

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 η -expansion for \rightarrow , then for $+$, then for `bool`.

4 Equality

λ -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 Γ 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:

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash M = M : B} \qquad \frac{\Gamma \vdash M = N : B}{\Gamma \vdash N = M : B}$$

$$\frac{\Gamma \vdash M = N : B \quad \Gamma \vdash N = P : B}{\Gamma \vdash M = P : B}$$

Secondly, we need rules to say that this is *compatible*—preserved by every construct:

$$\frac{\Gamma \vdash M = M' : A \quad \Gamma, \mathbf{x} : A \vdash N = N' : B}{\Gamma \vdash \text{let } M \text{ be } \mathbf{x}. N = \text{let } M' \text{ be } \mathbf{x}. N' : B}$$

and so forth. A compatible equivalence relation is often called a *congruence*.

Thirdly, each of the β -reductions that we've seen is an axiom of this theory.

$$\frac{\Gamma \vdash N : B \quad \Gamma \vdash N' : B}{\Gamma \vdash \text{match true as } \{\text{true. } N, \text{false. } N'\} = N : B}$$

$$\frac{\Gamma, \mathbf{x} : A \vdash M : B \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda \mathbf{x}. M)N = M[N/\mathbf{x}] : B}$$

Fourthly, each of the η -expansions is an axiom of the theory, e.g.

$$\frac{\Gamma \vdash M : A \rightarrow B}{\Gamma \vdash M = \lambda \mathbf{x}. (M\mathbf{x}) : A \rightarrow B}$$

But in the case of the η -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 η -law for `bool` looks like this:

$$\frac{\Gamma \vdash M : \text{bool} \quad \Gamma, z : \text{bool} \vdash N : C}{\Gamma \vdash N[M/z] = \text{match } M \text{ as } \{\text{true}.N[\text{true}/z], \text{false}.N[\text{false}/z]\} : C}$$

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

$$\begin{aligned} & \Gamma \vdash \text{match } M \text{ as } \{ \\ & \quad \text{true. match } M \text{ as } \{\text{true}.N_0, \text{false}.N_1\}, \\ & \quad \text{false. match } M \text{ as } \{\text{true}.N_2, \text{false}.N_3\} \\ & \quad \} \\ & = \text{match } M \text{ as } \{\text{true}.N_0, \text{false}.N_3\} : C \end{aligned}$$

2. Show that `inl` – is injective, i.e. if $\Gamma \vdash M, M' : A$ and $\Gamma \vdash \text{inl } M = \text{inl } M' : A + B$ then $\Gamma \vdash M = M' : A$.
3. Write down the η -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

$$\begin{aligned} & \Gamma, y : 0 \vdash M[N/x] = y : 0 \\ & \Gamma, x : A \vdash N[M/x] = x : A \end{aligned}$$

5. Give β and η 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.)