Polymorphism

Stefan Muller

CS 534: Types and Programming Languages, Spring 2024
Lecture 17

1 Why?

Consider the following functions:

$$\begin{array}{c}\lambda x : \mathsf{unit}.x \\ \lambda x : \mathsf{unit} \to \mathsf{unit}.x \\ \lambda x : \mathsf{int}.x \\ \lambda x : \mathsf{int} \times \mathsf{int}.x\end{array}$$

They all, of course, do the same thing: simply return their argument. In other words, they are all the identity function, but they don't have the same type. If we don't need type annotations, e.g., in the untyped lambda calculus, we can capture this behavior with the function $\lambda x.x$. But what is the type of this function?

2 Adding Polymorphism to STLC

$$\begin{array}{rcl}\tau & ::= & \cdots \mid \alpha \mid \forall \alpha.\tau \\ e & ::= & \cdots \mid \Lambda \alpha.e \mid e[\tau]\end{array}$$

We will be using type variables α like we did for recursive types, but now there's a new way to bind them: $\forall \alpha.\tau$. This represents the type of something that can have the type τ with α replaced by any type. We can also now use types in expressions! The expression $\Lambda \alpha.e$ is a function that takes a type as an argument, but it's an expression! We apply it to a type with $e[\tau]$.

Example: Polymorphic identity function. The identity function we want has the type

$$\forall \alpha.\alpha \to \alpha$$

It has the type $\alpha \to \alpha$ for any α. The identity function itself is:

$$\mathsf{id} \triangleq \Lambda \alpha.\lambda x : \alpha.x : \forall \alpha.\alpha \to \alpha$$

If we want to use it, we first need to give it a type:

$$\begin{array}{rcl}\mathsf{id}[\mathsf{unit}] & : & \mathsf{unit} \to \mathsf{unit} \\ \mathsf{id}[\mathsf{int}] & : & \mathsf{int} \to \mathsf{int} \\ \mathsf{id}[\mathsf{int} \times \mathsf{int}] & : & (\mathsf{int} \times \mathsf{int}) \to (\mathsf{int} \times \mathsf{int})\end{array}$$

In fact, doing so gives us the same specific identity functions we had before:

$$\begin{array}{rcl}\mathsf{id}[\mathsf{unit}] & \mapsto & \lambda x : \mathsf{unit}.x \\ \mathsf{id}[\mathsf{int}] & \mapsto & \lambda x : \mathsf{int}.x \\ \mathsf{id}[\mathsf{int} \times \mathsf{int}] & \mapsto & \lambda x : \mathsf{int} \times \mathsf{int}.x\end{array}$$

We again need to say when a type is well-formed:

$$\frac{\alpha \in \Delta}{\Delta \vdash \alpha\ \mathsf{ok}} \qquad\qquad \frac{\Delta, \alpha \vdash \tau\ \mathsf{ok}}{\Delta \vdash \forall\alpha.\tau\ \mathsf{ok}}$$

And, of course, the new expressions need statics and dynamics. The typing judgment now also needs the type variable context Δ because type variables can appear in expressions.

$$\frac{\Delta, \alpha; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau}\ (\forall\text{-I}) \qquad\qquad \frac{\Delta; \Gamma \vdash e : \forall\alpha.\tau \qquad \Delta \vdash \tau'\ \mathsf{ok}}{\Delta; \Gamma \vdash e[\tau'] : [\tau'/\alpha]\tau}\ (\forall\text{-E})$$

The dynamics are relatively easy:

$$\frac{}{\Lambda\alpha.e\ \mathsf{val}} \qquad\qquad \frac{e \mapsto e'}{e[\tau] \mapsto e'[\tau]} \qquad\qquad \frac{}{(\Lambda\alpha.e)[\tau] \mapsto [\tau/\alpha]e}$$

The last step rule justifies the steps we have above for the identity functions.

Example. The function $\Lambda\alpha.\forall\beta.\lambda x : \alpha \times \beta.(\mathsf{fst}\ x, \mathsf{snd}\ x)$ has type $\forall\alpha.\forall\beta.(\alpha \times \beta) \to (\beta \times \alpha)$.

$$\begin{array}{ll}
 & (\Lambda\alpha.\Lambda\beta.\lambda x : \alpha \times \beta.(\mathsf{fst}\ x, \mathsf{snd}\ x))[\mathsf{unit}][\mathsf{int}]\ ((), \overline{2}) \\
\mapsto & (\Lambda\beta.\lambda x : \mathsf{unit} \times \beta.(\mathsf{fst}\ x, \mathsf{snd}\ x))[\mathsf{int}]\ ((), \overline{2}) \\
\mapsto & (\lambda x : \mathsf{unit} \times \mathsf{int}.(\mathsf{fst}\ x, \mathsf{snd}\ x))\ ((), \overline{2}) \\
\mapsto & (\mathsf{fst}\ ((), \overline{2}), \mathsf{snd}\ ((), \overline{2})) \\
\mapsto & ((), \mathsf{snd}\ ((), \overline{2})) \\
\mapsto & ((), \overline{2})
\end{array}$$

3 Metatheory

Lemma 1 (Substitution for types). *If* $\Delta, \alpha \vdash \tau\ \mathsf{ok}$ *and* $\Delta \vdash \tau'\ \mathsf{ok}$ *then* $\Delta \vdash [\tau'/\alpha]\tau\ \mathsf{ok}$.

Proof. By induction on the derivation of $\Delta, \alpha \vdash \tau\ \mathsf{ok}$.

- If $\tau = \alpha$ then $[\tau'/\alpha]\tau = \tau'$. By assumption.
- If $\tau = \beta$ and $\beta \neq \alpha$ then $[\tau'/\alpha]\tau = \beta$. We have $\Delta \vdash \beta\ \mathsf{ok}$.
- If $\tau = \forall\beta.\tau''$ and (by alpha conversion) $\beta \neq \alpha$ and β is not free in τ', then $[\tau'/\alpha]\tau = \forall\beta.[\tau'/\alpha]\tau''$ and $\Delta, \alpha, \beta \vdash \tau''\ \mathsf{ok}$. By weakening, $\Delta, \beta \vdash \tau'\ \mathsf{ok}$. By induction, $\Delta, \beta \vdash [\tau'/\alpha]\tau''\ \mathsf{ok}$. We have $\Gamma \vdash \forall\beta.[\tau'/\alpha]\tau''\ \mathsf{ok}$.

□

In the below, we use $[\tau/\alpha]\Gamma$ to mean Γ with τ substituted for α in all of its types.

Lemma 2 (Substitution of types in expressions). *If* $\Delta, \alpha; \Gamma \vdash e : \tau$ *and* $\Delta \vdash \tau'\ \mathsf{ok}$ *then* $\Delta; [\tau'/\alpha]\Gamma \vdash [\tau'/\alpha]e : [\tau'/\alpha]\tau$.

Proof. By induction on the derivation of $\Delta, \alpha; \Gamma \vdash e : \tau$.

- $\to$-I. Then $e = \lambda x : \tau_0.e_0$ and $\tau = \tau_0 \to \tau_1$ and $\Delta, \alpha; \Gamma, x : \tau_0 \vdash e_0 : \tau_1$. We have $[\tau'/\alpha]e = \lambda x : [\tau'/\alpha]\tau_0.[\tau'/\alpha]e_0$ and $[\tau'/\alpha]\tau = ([\tau'/\alpha]\tau_0) \to ([\tau'/\alpha]\tau_1)$. By induction, $\Delta; [\tau'/\alpha]\Gamma, x : [\tau'/\alpha]\tau_0 \vdash [\tau'/\alpha]e_0 : [\tau'/\alpha]\tau_1$. Apply rule $\to$-I.
- $\forall$-E. Then $e = e_0[\tau'']$ and $\Delta; \Gamma \vdash e_0 : \forall\beta.\tau_0$ (where we can assume $\beta \neq \alpha$) and $\Delta \vdash \tau''\ \mathsf{ok}$ and $\tau = [\tau''/\beta]\tau_0$. We have $[\tau'/\alpha]e = ([\tau'/\alpha]e_0)[[\tau'/\alpha]\tau'']$ and $[\tau'/\alpha]\tau = [[\tau'/\alpha]\tau''/\beta][\tau'/\alpha]\tau_0$. By induction, $\Delta; [\tau'/\alpha]\Gamma \vdash [\tau'/\alpha]e_0 : \forall\beta.[\tau'/\alpha]\tau_0$. By Lemma 1, $\Delta \vdash [\tau'/\alpha]\tau''\ \mathsf{ok}$. Apply rule $\forall$-E.

□

Below, $\Delta \vdash \Gamma$ ok is defined to mean that $\Delta \vdash \tau$ ok for every τ such that $x : \tau \in \Gamma$.

Lemma 3. *If $\Delta; \Gamma \vdash e : \tau$ and $\Delta \vdash \Gamma$ ok then $\Delta \vdash \tau$ ok.*

Proof. By induction on the derivation of $\Delta; \Gamma \vdash e : \tau$.

- VAR. Then $e = x$ and $\Gamma(x) = \tau$. Because $\Delta \vdash \Gamma$ ok, we have $\Delta \vdash \tau$ ok.
- $\forall$-I. Then $\tau = \forall \alpha.\tau'$ and $e = \Lambda\alpha.e'$ and $\Delta, \alpha; \Gamma \vdash e : \tau'$. By induction, $\Delta, \alpha \vdash \tau'$ ok. We then have $\Delta \vdash \tau$ ok.
- $\forall$-E. Then $e = e'[\tau']$ and $\Delta; \Gamma \vdash e' : \forall\alpha.\tau''$ and $\tau = [\tau'/\alpha]\tau''$ and $\Delta \vdash \tau'$ ok. By induction, $\Delta \vdash \forall\alpha.\tau''$ ok. By inversion, $\Delta, \alpha \vdash \tau''$ ok. By Lemma 1, $\Delta \vdash \tau$ ok.

□

Lemma 4 (Progress)**.** *If $\bullet; \bullet \vdash e : \tau$ then either e val or there exists e' such that $e \mapsto e'$.*

Proof. By induction on the derivation of $\Delta; \Gamma \vdash e : \tau$.

- $\forall$-I. Then $e = \Lambda\alpha.e'$ and we have e val.
- $\forall$-E. Then $e = e_0[\tau']$ and $\Delta; \Gamma \vdash e_0 : \forall\alpha.\tau''$ and $\tau = [\tau'/\alpha]\tau''$ and $\Delta \vdash \tau'$ ok. By induction, e_0 val or $e_0 \mapsto e_0'$.
  - e_0 val. Then by canonical forms, $e_0 = \Lambda\alpha.e_0'$ and $e \mapsto [\tau'/\alpha]e_0'$.
  - $e_0 \mapsto e_0'$. Then $e \mapsto e_0'[\tau']$.

□

Lemma 5 (Preservation)**.** *If $\bullet; \bullet \vdash e : \tau$ and $e \mapsto e'$ then $\bullet; \bullet \vdash e' : \tau$.*

Proof. By induction on the derivation of $e \mapsto e'$.

- $\dfrac{}{(\Lambda\alpha.e_0)[\tau'] \mapsto [\tau'/\alpha]e_0}$ By inversion on $\forall$-E, we have $\bullet; \bullet \vdash \Lambda\alpha.e_0 : \forall\alpha.\tau''$ and $\bullet \vdash \tau'$ ok and $\tau = [\tau'/\alpha]\tau''$. By inversion on $\forall$-I, we have $\alpha; \bullet \vdash e_0 : \tau''$. By Lemma 2, we have $\bullet; \bullet \vdash [\tau'/\alpha]e_0 : [\tau'/\alpha]\tau''$.

□