Polymorphism

CS 152 (Spring 2020)

Harvard University

Tuesday, March 24, 2020

Today, we will learn about

- Parametric Polymorphism
- Subtyping

Polymorphism

- Subtype polymorphism
- Ad-hoc polymorphism
- Parametric polymorphism

Parametric Polymorphism in STLC?

$$\begin{aligned}
\mathit{doubleInt} &\triangleq \lambda f : \mathbf{int} \to \mathbf{int}.\ \lambda x : \mathbf{int}.\ f\ (f\ x) \\
\mathit{doubleBool} &\triangleq \lambda f : \mathbf{bool} \to \mathbf{bool}.\ \lambda x : \mathbf{bool}.\ f\ (f\ x) \\
\mathit{doubleFn} &\triangleq \lambda f : (\mathbf{int} \to \mathbf{int}) \to (\mathbf{int} \to \mathbf{int}). \\
&\qquad \lambda x : \mathbf{int} \to \mathbf{int}.\ f\ (f\ x) \\
&\ \ \vdots
\end{aligned}$$

Abstraction Principle

Each significant piece of functionality in a program should be implemented in just one place in the source code. When similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts.

Type Abstraction in System F

A *type abstraction* is a new expression, written $\Lambda X.\, e$, where Λ is the upper-case form of the Greek letter lambda, and X is a *type variable*. We also introduce a new form of application, called *type application*, or *instantiation*, written $e_1\ [\tau]$.

Syntax of System F

$$e ::= n \mid x \mid \lambda x : \tau.\, e \mid e_1\ e_2 \mid \Lambda X.\, e \mid e\ [\tau]$$
$$v ::= n \mid \lambda x : \tau.\, e \mid \Lambda X.\, e$$

Operational Semantics of System F

$$E ::= [\cdot] \mid E\ e \mid v\ E \mid E\ [\tau]$$

$$\frac{e \longrightarrow e'}{E[e] \longrightarrow E[e']}$$

$$\beta\text{-reduction}\ \frac{}{(\lambda x{:}\tau.\, e)\ v \longrightarrow e\{v/x\}}$$

$$\textsc{Type-reduction}\ \frac{}{(\Lambda X.\, e)\ [\tau] \longrightarrow e\{\tau/X\}}$$

Example (Polymorphic Identity Function)

$$ID \triangleq \Lambda X.\, \lambda x : X.\, x$$

$$(\Lambda X.\, \lambda x : X.\, x)\ [\mathbf{int}] \longrightarrow \lambda x : \mathbf{int}.\, x$$

$$(\Lambda X.\, \lambda x : X.\, x)\ [\mathbf{int} \to \mathbf{int}] \longrightarrow \lambda x : \mathbf{int} \to \mathbf{int}.\, x$$

Type System of System F (Syntax)

$$\tau ::= \mathbf{int} \mid \tau_1 \to \tau_2 \mid X \mid \forall X.\ \tau$$

Type System of System F (Well-Typed)

$$\frac{}{\Delta, \Gamma \vdash n : \mathbf{int}} \qquad \frac{\Delta \vdash \tau \text{ ok}}{\Delta, \Gamma \vdash x : \tau}\, \Gamma(x) = \tau$$

$$\frac{\Delta, \Gamma, x : \tau \vdash e : \tau' \quad \Delta \vdash \tau \text{ ok}}{\Delta, \Gamma \vdash \lambda x : \tau.\, e : \tau \to \tau'}$$

$$\frac{\Delta, \Gamma \vdash e_1 : \tau \to \tau' \quad \Delta, \Gamma \vdash e_2 : \tau}{\Delta, \Gamma \vdash e_1\ e_2 : \tau'}$$

$$\frac{\Delta \cup \{X\}, \Gamma \vdash e : \tau}{\Delta, \Gamma \vdash \Lambda X.\, e : \forall X.\, \tau}$$

$$\frac{\Delta, \Gamma \vdash e : \forall X.\, \tau' \quad \Delta \vdash \tau \text{ ok}}{\Delta, \Gamma \vdash e\ [\tau] : \tau'\{\tau/X\}}$$

Type System of System F (Well-Formed)

$$\frac{}{\Delta \vdash X \text{ ok}}\, X \in \Delta \qquad \frac{}{\Delta \vdash \textbf{int} \text{ ok}}$$

$$\frac{\Delta \vdash \tau_1 \text{ ok} \quad \Delta \vdash \tau_2 \text{ ok}}{\Delta \vdash \tau_1 \to \tau_2 \text{ ok}} \qquad \frac{\Delta \cup \{X\} \vdash \tau \text{ ok}}{\Delta \vdash \forall X.\, \tau \text{ ok}}$$

Examples

$$\mathit{double} \triangleq \Lambda X.\, \lambda f : X \to X.\, \lambda x : X.\, f\ (f\ x).$$

$$\forall X.\, (X \to X) \to X \to X$$

$$\begin{aligned}
&\mathit{double}\ [\mathbf{int}]\ (\lambda n : \mathbf{int}.\, n + 1)\ 7 \\
&\longrightarrow (\lambda f : \mathbf{int} \to \mathbf{int}.\, \lambda x : \mathbf{int}.\, f\ (f\ x))\ (\lambda n : \mathbf{int}.\, n + 1)\ 7 \\
&\longrightarrow^{*} 9
\end{aligned}$$

Example: Self-Application

$$\begin{aligned} \vdash \quad & \lambda x : \forall X.\ X \to X.\ x\ [\forall X.\ X \to X]\ x \\ : \quad & (\forall X.\ X \to X) \to (\forall X.\ X \to X) \end{aligned}$$

Records (Syntax)

$$
\begin{aligned}
& l \in \mathcal{L} \\
& e ::= \cdots \mid \{l_1 = e_1, \ldots, l_n = e_n\} \mid e.l \\
& v ::= \cdots \mid \{l_1 = v_1, \ldots, l_n = v_n\} \\
& \tau ::= \cdots \mid \{l_1 : \tau_1, \ldots, l_n : \tau_n\}
\end{aligned}
$$

Records (Operational Semantics)

$$
\begin{aligned}
E ::= & \ldots \\
& \mid \{l_1 = v_1, \ldots, l_i = E, \ldots, l_n = e_n\} \\
& \mid E.l
\end{aligned}
$$

$$
\frac{}{\{l_1 = v_1, \ldots, l_n = v_n\}.l_i \longrightarrow v_i}
$$

Records (Typing)

$$\frac{\forall i \in 1..n. \quad \Gamma \vdash e_i : \tau_i}{\Gamma \vdash \{l_1 = e_1, \ldots, l_n = e_n\} : \{l_1 : \tau_1, \ldots, l_n : \tau_n\}}$$

$$\frac{\Gamma \vdash e : \{l_1 : \tau_1, \ldots, l_n : \tau_n\}}{\Gamma \vdash e.l_i : \tau_i}$$

Subtyping (Principle)

The principle of subtyping is as follows. If τ_1 *is a subtype of* τ_2 *(written* $\tau_1 \leq \tau_2$*, and also sometimes as* $\tau_1 \leq: \tau_2$*), then a program can use a value of type* τ_1 *whenever it would use a value of type* τ_2*. If* $\tau_1 \leq \tau_2$*, then* τ_1 *is sometimes referred to as the subtype, and* τ_2 *as the supertype.*

Principle of Subtyping in Typing

$$\textsc{Subsumption}\ \frac{\Gamma \vdash e : \tau \quad \tau \leq \tau'}{\Gamma \vdash e : \tau'}$$

Subtyping Properties

$$\frac{}{\tau \leq \tau} \qquad \frac{\tau_1 \leq \tau_2 \quad \tau_2 \leq \tau_3}{\tau_1 \leq \tau_3}$$

Subtyping of Records

$$\frac{\forall i \in 1..n.\ \exists j \in 1..m. \quad l'_i = l_j \quad \wedge \quad \tau_j \leq \tau'_i}{\{l_1 : \tau_1, \ldots, l_m : \tau_m\} \leq \{l'_1 : \tau'_1, \ldots, l'_n : \tau'_n\}}$$

Subtyping of Products

$$\frac{\tau_1 \leq \tau_1' \quad \tau_2 \leq \tau_2'}{\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'}$$

Subtyping of Functions

$$\frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}$$

Subtyping of Locations

$$\frac{\tau \leq \tau' \qquad \tau' \leq \tau}{\tau \ \mathbf{ref} \leq \tau' \ \mathbf{ref}}$$