

Lectures #18–19: Typed λ -calculi

CS 6371: Advanced Programming Languages

October 23 and 28, 2008

There are many typed λ -calculi, of which we will study two. The first is generally referred to as the *simply-typed λ -calculus* or λ^\rightarrow . Here are some of the operations commonly included in λ^\rightarrow .

$e ::= n$	integers
v	variables
$\lambda v:\tau. e$	abstraction
$e_1 e_2$	application
true false	booleans
$e_1 \text{ aop } e_2$	arithmetic ops
$e_1 \text{ bop } e_2$	boolean ops
$e_1 \text{ cmp } e_2$	integer comparison
(e_1, e_2)	pairs
$\pi_1 e$ $\pi_2 e$	projection
$()$	unit
$\mathbf{in}_1^{\tau_1+\tau_2} e$ $\mathbf{in}_2^{\tau_1+\tau_2} e$	injection
$(\mathbf{case } e \text{ of } \mathbf{in}_1(v_1) \rightarrow e_1 \mid \mathbf{in}_2(v_2) \rightarrow e_2)$	case distinction

The type system for the above language is as follows.

$\tau ::= \mathit{int}$	integer
bool	boolean
$\tau_1 \rightarrow \tau_2$	function
$\tau_1 \times \tau_2$	product type
unit	unit type
$\tau_1 + \tau_2$	sum type
void	void type

$$\begin{array}{lcl}
\Gamma \vdash n : \mathit{int} & (1) & \\
\Gamma \vdash v : \Gamma(v) & (2) & \\
\frac{\Gamma[v \mapsto \tau_1] \vdash e : \tau_2}{\Gamma \vdash (\lambda v : \tau. e) : \tau_1 \rightarrow \tau_2} & (3) & \\
\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} & (4) & \\
\Gamma \vdash \mathbf{true} : \mathit{bool} & (5) & \\
\Gamma \vdash \mathbf{false} : \mathit{bool} & (6) & \\
\frac{\Gamma \vdash e_1 : \mathit{int} \quad \Gamma \vdash e_2 : \mathit{int}}{\Gamma \vdash e_1 \mathit{aop} e_2 : \mathit{int}} & (7) & \\
\frac{\Gamma \vdash e_1 : \mathit{bool} \quad \Gamma \vdash e_2 : \mathit{bool}}{\Gamma \vdash e_1 \mathit{bop} e_2 : \mathit{bool}} & (8) & \\
\frac{\Gamma \vdash e_1 : \mathit{int} \quad \Gamma \vdash e_2 : \mathit{int}}{\Gamma \vdash e_1 \mathit{cmp} e_2 : \mathit{bool}} & (9) & \\
\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} & (10) & \\
\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \quad i \in \{1, 2\}}{\Gamma \vdash \pi_i e : \tau_i} & (11) & \\
\Gamma \vdash () : \mathit{unit} & (12) & \\
\frac{\Gamma \vdash e : \tau_i \quad i \in \{1, 2\}}{\Gamma \vdash \mathbf{in}_i^{\tau_1 + \tau_2} e : \tau_1 + \tau_2} & (13) & \\
\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma[v_1 \mapsto \tau_1] \vdash e_1 : \tau \quad \Gamma[v_2 \mapsto \tau_2] \vdash e_2 : \tau}{\Gamma \vdash (\mathbf{case} e \mathbf{of} \mathbf{in}_1(v_1) \rightarrow e_1 \mid \mathbf{in}_2(v_2) \rightarrow e_2) : \tau} & (14) &
\end{array}$$

The simply-typed λ -calculus is often extended with an explicit fixed-point operator, sometimes notated with the letter μ as follows.

$$\begin{array}{l}
e ::= \dots \mid \mu v : \tau. e \\
\mu v : \tau. e \rightarrow_1 e[(\mu v : \tau. e)/v] \\
\frac{\Gamma[v \mapsto \tau] \vdash e : \tau}{\Gamma \vdash (\mu v : \tau. e) : \tau} \quad (15)
\end{array}$$

System F [Girard 1972, Reynolds 1974], also known as the *polymorphic λ -calculus*, extends the simply-typed λ -calculus with *type variables*. Valid System F expressions contain no free variables and no free type variables.

$$\begin{array}{lcl}
e ::= \dots \mid \Lambda \alpha. e & & \text{polymorphic abstraction} \\
& \mid e[\tau] & \text{polymorphic instantiation} \\
\tau ::= \dots \mid \alpha & & \text{type variables} \\
& \mid \forall \alpha. \tau & \text{universal types}
\end{array}$$

$$(\Lambda \alpha. e)[\tau] \rightarrow_1 e[\tau/\alpha]$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \Lambda \alpha. e : \forall \alpha. \tau} \quad (16)$$

$$\frac{\Gamma \vdash e : \forall \alpha. \tau'}{\Gamma \vdash e[\tau] : \tau'[\tau/\alpha]} \quad (17)$$