COSE212: Programming Languages

Lecture 12 — Automatic Type Inference (3)

Hakjoo Oh

2016 Fall

Type Inference for PROC

$$\boxed{\textbf{typeof} : E \rightarrow T}$$

$$
\begin{array}{rcl}
E & \rightarrow & n \\
 & | & x \\
 & | & E + E \\
 & | & E - E \\
 & | & \texttt{iszero}\ E \\
 & | & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
 & | & \texttt{let}\ x = E\ \texttt{in}\ E \\
 & | & \texttt{proc}\ x\ E \\
 & | & E\ E \\
 & & \\
T & \rightarrow & \mathsf{int} \\
 & | & \mathsf{bool} \\
 & | & T \rightarrow T \\
 & | & \alpha\ (\in \mathit{TyVar})
\end{array}
$$

Deriving Type Equations

- Type equations:

$$TyEqn \to \emptyset \mid T \doteq T \ \wedge \ TyEqn$$

- Generation algorithm:

$$\mathcal{V} : (Id \to T) \times E \times T \to TyEqn$$

$\mathcal{V}(\Gamma, e, t)$ generates constraint u such that

$$\Gamma \vdash e : t$$

is true if u is satisfied.

  - $\mathcal{V}([x \mapsto \mathsf{int}], \texttt{x+1}, \alpha) =$
  - $\mathcal{V}(\emptyset, \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ 1\ \texttt{else}\ 2), \alpha \to \beta) =$

Deriving Type Equations

$$
\begin{array}{rcl}
\mathcal{V}(\Gamma, n, t) & = & t \doteq \mathsf{int} \\
\mathcal{V}(\Gamma, x, t) & = & t \doteq \Gamma(x) \\
\mathcal{V}(\Gamma, e_1 + e_2, t) & = & t \doteq \mathsf{int} \ \wedge\ \mathcal{V}(\Gamma, e_1, \mathsf{int}) \ \wedge\ \mathcal{V}(\Gamma, e_2, \mathsf{int}) \\
\mathcal{V}(\Gamma, \mathtt{iszero}\ e, t) & = & t \doteq \mathsf{bool} \ \wedge\ \mathcal{V}(\Gamma, e, \mathsf{int}) \\
\mathcal{V}(\Gamma, \mathtt{if}\ e_1\ e_2\ e_3, t) & = & \mathcal{V}(\Gamma, e_1, \mathsf{bool}) \ \wedge\ \mathcal{V}(\Gamma, e_2, t) \ \wedge\ (\Gamma, e_3, t) \\
\mathcal{V}(\Gamma, \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2, t) & = & \mathcal{V}(\Gamma, e_1, \alpha) \ \wedge\ \mathcal{V}([x \mapsto \alpha]\Gamma, e_2, t)\ (\text{new } \alpha) \\
\mathcal{V}(\Gamma, \mathtt{proc}\ (x)\ e, t) & = & t \doteq \alpha_1 \to \alpha_2 \ \wedge\ \mathcal{V}([x \mapsto \alpha_1]\Gamma, e, \alpha_2) \\
 & & (\text{new } \alpha_1, \alpha_2) \\
\mathcal{V}(\Gamma, e_1\ e_2, t) & = & \mathcal{V}(\Gamma, e_1, \alpha \to t) \ \wedge\ \mathcal{V}(\Gamma, e_2, \alpha)\ (\text{new } \alpha)
\end{array}
$$

Exercises

- $\mathcal{V}(\emptyset, (\texttt{proc}\ (x)\ (x))\ 1, \alpha)$
- $\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (f\ 11), \alpha)$
- $\mathcal{V}([x \mapsto \mathsf{bool}], \texttt{if}\ x\ \texttt{then}\ (x-1)\ \texttt{else}\ 0, \alpha)$
- $\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (\texttt{iszero}\ (f\ f)), \alpha)$

Substitution

Solutions of type equations are represented by substitution:

$$S \in Subst = TyVar \to T$$

Applying a substitution to a type:

$$\begin{array}{rcl} S(\text{int}) & = & \text{int} \\ S(\text{bool}) & = & \text{bool} \\ S(\alpha) & = & \begin{cases} t & \text{if } \alpha \mapsto t \in S \\ \alpha & \text{otherwise} \end{cases} \\ S(T_1 \to T_2) & = & S(T_1) \to S(T_2) \end{array}$$

Unification

Update the current substitution with equality $t_1 \doteq t_2$.

$$\textbf{unify} : T \times T \times Subst \rightarrow Subst$$

$$
\begin{array}{rcl}
\textbf{unify}(\text{int}, \text{int}, S) & = & S \\
\textbf{unify}(\text{bool}, \text{bool}, S) & = & S \\
\textbf{unify}(\alpha, t, S) & = & \begin{cases} \text{fail} & \alpha \text{ occurs in } t \\ \text{extend } S \text{ with } \alpha \doteq t & \text{otherwise} \end{cases} \\
\textbf{unify}(t, \alpha, S) & = & \textbf{unify}(\alpha, t, S) \\
\textbf{unify}(t_1 \rightarrow t_2, t'_1 \rightarrow t'_2, S) & = & \text{let } S' = \textbf{unify}(t_1, t'_1, S) \text{ in} \\
 & & \text{let } t_3 = S'(t_2) \text{ in} \\
 & & \text{let } t_4 = S'(t'_2) \text{ in} \\
 & & \quad \textbf{unify}(t_3, t_4, S') \\
\textbf{unify}(\_, \_, \_) & = & \text{fail}
\end{array}
$$

cf) extension of S with $\alpha \doteq t$:

$$[\alpha \mapsto t]\{\alpha_1 \mapsto \{\alpha \mapsto t\}(t_1) \mid \alpha_1 \mapsto t_1 \in S\}$$

Exercises

- $\textbf{unify}(\alpha, \text{int} \to \text{int}, \emptyset) =$
- $\textbf{unify}(\alpha, \text{int} \to \alpha, \emptyset) =$
- $\textbf{unify}(\alpha \to \beta, \text{int} \to \text{int}, \emptyset) =$
- $\textbf{unify}(\alpha \to \beta, \text{int} \to \alpha, \emptyset) =$

Solving Equations

$$\textbf{unifyall} : TyEqn \to Subst \to Subst$$

$$\begin{array}{rcl} \textbf{unifyall}(\emptyset, S) & = & S \\ \textbf{unifyall}((t_1 \doteq t_2) \ \land \ u, S) & = & \text{let } S' = \textbf{unify}(S(t_1), S(t_2), S) \\ & & \text{in } \textbf{unifyall}(u, S') \end{array}$$

typeof

$$
\begin{array}{l}
\textbf{typeof}(E) = \\
\quad \textbf{let } S = \mathcal{U}(\mathcal{V}(\emptyset, E, \alpha)) \quad (\text{new } \alpha) \\
\quad \textbf{in } S(\alpha)
\end{array}
$$

Examples

- **typeof**$((\texttt{proc}\ (x)\ x)\ 1)$
- **typeof**$(\texttt{let}\ x = 1\ \texttt{in}\ \texttt{proc}(y)\ (x + y))$

Summary: Automatic Type Inference

Design and implementation of static type system:

- logical rules for inferring types
- algorithmic procedure for inferring types