CVO103: Programming Languages

Lecture 13 — Automatic Type Inference (3)

Hakjoo Oh
2018 Spring

Putting It All Together

- So far we have informally discussed automatic type inference.
- In this lecture, we define the algorithm precisely.

Goal

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

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

Deriving Type Equations

- Type equations:

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

- Algorithm for generating equations:

$$\mathcal{V} : (Var \rightarrow T) \times E \times T \rightarrow TyEqn$$

- $\mathcal{V}(\Gamma, e, t)$ generates the condition for e to have type t in Γ:

$$\Gamma \vdash e : t \text{ iff } \mathcal{V}(\Gamma, e, t) \text{ is satisfied.}$$

  - $\mathcal{V}([x \mapsto \text{int}], \texttt{x+1}, \alpha) = \alpha \doteq \text{int}$
  - $\mathcal{V}(\emptyset, \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ 1\ \texttt{else}\ 2), \alpha \rightarrow \beta) =$
    $\alpha \doteq \text{bool} \wedge \beta \doteq \text{int}$

Deriving Type Equations

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

Example

$$
\begin{aligned}
& \mathcal{V}(\emptyset, (\texttt{proc}\ (x)\ (x))\ 1, \alpha) & \\
& = \mathcal{V}(\emptyset, \texttt{proc}\ (x)\ (x), \alpha_1 \to \alpha) \wedge \mathcal{V}(\emptyset, 1, \alpha_1) & \text{new } \alpha_1 \\
& = \alpha_1 \to \alpha \doteq \alpha_2 \to \alpha_3 \wedge \mathcal{V}([x \mapsto \alpha_2], x, \alpha_3) \wedge \alpha_1 \doteq \mathsf{int} & \text{new } \alpha_2, \alpha_3 \\
& = \alpha_1 \to \alpha \doteq \alpha_2 \to \alpha_3 \wedge \alpha_2 \doteq \alpha_3 \wedge \alpha_1 \doteq \mathsf{int} &
\end{aligned}
$$

Exercise 1

$$\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (f\ 11), \alpha)$$

Exercise 2

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

Exercise 3

$$\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (\texttt{iszero}\ (f\ f)), \alpha)$$

Substitution

Solutions of type equations are represented by substitution:

$$S \in \mathit{Subst} = \mathit{TyVar} \to T$$

Applying a substitution to a type:

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

Example

Applying the substitution

$$S = \{t_1 \mapsto \mathsf{int}, t_2 \mapsto \mathsf{int} \to \mathsf{int}\}$$

to to the type $(t_1 \to t_2) \to (t_3 \to \mathsf{int})$:

$$\begin{aligned}
& S((t_1 \to t_2) \to (t_3 \to \mathsf{int})) \\
&= S(t_1 \to t_2) \to S(t_3 \to \mathsf{int}) \\
&= (S(t_1) \to S(t_2)) \to (S(t_3) \to S(\mathsf{int})) \\
&= (\mathsf{int} \to (\mathsf{int} \to \mathsf{int})) \to (t_3 \to \mathsf{int})
\end{aligned}$$

Unification

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

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

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

Exercises

- $\mathbf{unify}(\alpha, \mathsf{int} \to \mathsf{int}, \emptyset) =$
- $\mathbf{unify}(\alpha, \mathsf{int} \to \alpha, \emptyset) =$
- $\mathbf{unify}(\alpha \to \beta, \mathsf{int} \to \mathsf{int}, \emptyset) =$
- $\mathbf{unify}(\alpha \to \beta, \mathsf{int} \to \alpha, \emptyset) =$

Solving Equations

$$\textbf{unifyall} : \mathit{TyEqn} \rightarrow \mathit{Subst} \rightarrow \mathit{Subst}$$

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

Let $\mathcal{U}$ be the final unification algorithm:

$$\mathcal{U}(u) = \textbf{unifyall}(u, \emptyset)$$

typeof : $E \to T$

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

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