CVO103: Programming Languages

Lecture 11 — Automatic Type Inference (1)

Hakjoo Oh 2018 Spring

The Problem of Automatic Type Inference

Given a program E, infer the most general type of E if E can be typed (i.e., $[] \vdash E : t$ for some $t \in T$). If E cannot be typed, say so.

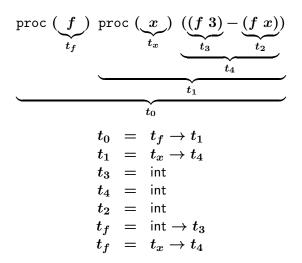
- let $f = \operatorname{proc}(x)(x+1)$ in $(\operatorname{proc}(x)(x1)) f$
- let f = proc (x) (x + 1) in (proc (x) (x true)) f
- ullet proc (x) x

Automatic Type Inference

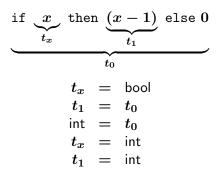
- A static analysis algorithm that automatically figures out types of expressions by observing how they are used.
- The algorithm is *sound and complete* with respect to the type system design.
 - ▶ (Sound) If the analysis finds a type for an expression, the expression is well-typed with the type according to the type system.
 - (Complete) If an expression has a type according to the type system, the analysis is guaranteed to find the type.
- The algorithm consists of two steps:
 - Generate type equations from the program text.
 - Solve the equations.

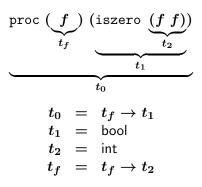
Generating Type Equations

For every subexpression and variable, introduce type variables and derive equations between the type variables.



$$t_0 = t_f o t_1 \ t_1$$





Idea: Deriving Equations from Typing Rules

For each expression e and variable x, let t_e and t_x denote the type of the expression and variable. Then, the typing rules dictate the equations that must hold between the type variables.

$$egin{array}{cccc} rac{\Gamma dash E_1: ext{int}}{\Gamma dash E_1 + E_2: ext{int}} \ & t_{E_1} = ext{int} \ & t_{E_2} = ext{int} \ & t_{E_1 + E_2} = ext{int} \end{array}$$

$$oldsymbol{\Gamma} dash E: \mathsf{int} \ \Gamma dash \mathsf{iszero} \ E: \mathsf{bool}$$

$$t_E = \mathsf{int} \ \land \ t_{(\mathsf{iszero}\ E)} = \mathsf{bool}$$

$$egin{aligned} \Gamma dash E_1: t_1
ightarrow t_2 & \Gamma dash E_2: t_1 \ \Gamma dash E_1 \ E_2: t_2 & \ & t_{E_1} = t_{E_2}
ightarrow \ t_{(E_1 \ E_2)} \end{aligned}$$

Idea: Deriving Equations from Typing Rules

$$\begin{array}{lll} \Gamma \vdash E_1 : \mathsf{bool} & \Gamma \vdash E_2 : t & \Gamma \vdash E_3 : t \\ \hline \Gamma \vdash \mathsf{if} \ E_1 \ \mathsf{then} \ E_2 \ \mathsf{else} \ E_3 : t \\ & t_{E_1} &= \ \mathsf{bool} \ \land \\ & t_{E_2} &= \ t_{(\mathsf{if} \ E_1 \ \mathsf{then} \ E_2 \ \mathsf{else} \ E_3)} \ \land \\ & t_{E_3} &= \ t_{(\mathsf{if} \ E_1 \ \mathsf{then} \ E_2 \ \mathsf{else} \ E_3)} \end{array} \land \\ & \underbrace{ \begin{bmatrix} x \mapsto t_1 \end{bmatrix} \Gamma \vdash E : t_2 }_{\Gamma \vdash \mathsf{proc} \ x \ E : t_1 \to t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{proc} \ x \ E : t_1 \to t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{proc} \ x \ E : t_1 \to t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{proc} \ x \ E : t_1 \to t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{in} \ E_2 : t_2} \\ & \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{let} \ x = E_1 \ \mathsf{let} \ \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{let} \ \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{let} \ \underbrace{ \begin{matrix} (x \mapsto t_1) \Gamma \vdash E_2 : t_2 \end{matrix} }_{\Gamma \vdash \mathsf{let} \ x = E_1 \ \mathsf{let} \$$

Summary

The algorithm for automatic type inference:

- Generate type equations from the program text.
 - Introduce type variables for each subexpression and variable.
 - ► Generate equations between type variables according to typing rules.
- Solve the equations.