

COSE212: Programming Languages

Lecture 13 — Automatic Type Inference (1)

Hakjoo Oh
2022 Fall

The Problem of Automatic Type Inference

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

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

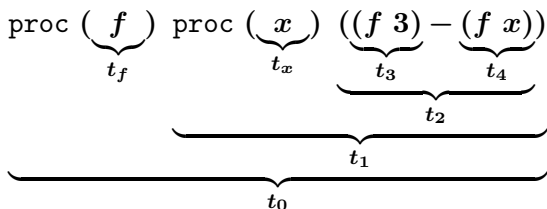
Automatic Type Inference

- 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:
 - 1 Generate type equations from the program text.
 - 2 Solve the equations.

Generating Type Equations

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

Example 1



$$\begin{array}{l} t_0 = t_f \rightarrow t_1 \\ t_1 = t_x \rightarrow t_4 \\ t_3 = \text{int} \\ t_4 = \text{int} \\ t_2 = \text{int} \\ t_f = \text{int} \rightarrow t_3 \\ t_f = t_x \rightarrow t_4 \end{array}$$

Example 2

$$\text{proc } \underbrace{(f)}_{t_f} \underbrace{(f \ 11)}_{t_1}$$
$$\underbrace{\hspace{10em}}_{t_0}$$

$$t_0 = t_f \rightarrow t_1$$

$$t_f = \text{int} \rightarrow t_1$$

Example 3

if \underbrace{x}_{t_x} then $\underbrace{(x - 1)}_{t_1}$ else 0

$\underbrace{\hspace{10em}}_{t_0}$

$t_x = \text{bool}$

$t_1 = t_0$

$\text{int} = t_0$

$t_x = \text{int}$

$t_1 = \text{int}$

Example 4

$\text{proc } \underbrace{(f)}_{t_f} \underbrace{(\text{iszero } \underbrace{(f f)}_{t_2})}_{t_1}$
 $\underbrace{\hspace{10em}}_{t_0}$

$t_0 = t_f \rightarrow t_1$

$t_1 = \text{bool}$

$t_2 = \text{int}$

$t_f = t_f \rightarrow t_2$

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.

$$\bullet \frac{\Gamma \vdash E_1 : \text{int} \quad \Gamma \vdash E_2 : \text{int}}{\Gamma \vdash E_1 + E_2 : \text{int}}$$

$$t_{E_1} = \text{int} \wedge t_{E_2} = \text{int} \wedge t_{E_1 + E_2} = \text{int}$$

$$\bullet \frac{\Gamma \vdash E : \text{int}}{\Gamma \vdash \text{iszero } E : \text{bool}}$$

$$t_E = \text{int} \wedge t_{(\text{iszero } E)} = \text{bool}$$

$$\bullet \frac{\Gamma \vdash E_1 : t_1 \rightarrow t_2 \quad \Gamma \vdash E_2 : t_1}{\Gamma \vdash E_1 E_2 : t_2}$$

$$t_{E_1} = t_{E_2} \rightarrow t_{(E_1 E_2)}$$

Idea: Deriving Equations from Typing Rules

$$\bullet \frac{\Gamma \vdash E_1 : \text{bool} \quad \Gamma \vdash E_2 : t \quad \Gamma \vdash E_3 : t}{\Gamma \vdash \text{if } E_1 \text{ then } E_2 \text{ else } E_3 : t}$$

$$t_{E_1} = \text{bool} \wedge$$

$$t_{E_2} = t_{(\text{if } E_1 \text{ then } E_2 \text{ else } E_3)} \wedge$$

$$t_{E_3} = t_{(\text{if } E_1 \text{ then } E_2 \text{ else } E_3)}$$

$$\bullet \frac{[x \mapsto t_1] \Gamma \vdash E : t_2}{\Gamma \vdash \text{proc } x \ E : t_1 \rightarrow t_2}$$

$$t_{(\text{proc } (x) \ E)} = t_x \rightarrow t_E$$

$$\bullet \frac{\Gamma \vdash E_1 : t_1 \quad [x \mapsto t_1] \Gamma \vdash E_2 : t_2}{\Gamma \vdash \text{let } x = E_1 \text{ in } E_2 : t_2}$$

$$t_x = t_{E_1} \wedge t_{E_2} = t_{(\text{let } x = E_1 \text{ in } E_2)}$$

Summary

The algorithm for automatic type inference:

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