# COSE212: Programming Languages

# Lecture 13 — Automatic Type Inference (1)

Hakjoo Oh

2018 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., $[] \vdash E : t$ for some $t \in T$). If $E$ cannot be typed, say so.

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

# 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

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ \underbrace{\texttt{proc}\ (\underbrace{x}_{t_x})\ \underbrace{(\underbrace{(f\ 3)}_{t_3} - \underbrace{(f\ x)}_{t_2})}_{t_4}}_{t_1}}_{t_0}$$

$$\begin{array}{rcl}
t_0 & = & t_f \to t_1 \\
t_1 & = & t_x \to t_4 \\
t_3 & = & \mathsf{int} \\
t_4 & = & \mathsf{int} \\
t_2 & = & \mathsf{int} \\
t_f & = & \mathsf{int} \to t_3 \\
t_f & = & t_x \to t_4
\end{array}$$

# Example 2

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ (\underbrace{f\ 11}_{t_1})}_{t_0}$$

$$\begin{array}{rcl} t_0 & = & t_f \to t_1 \\ t_f & = & \mathsf{int} \to t_1 \end{array}$$

# Example 3

$$\underbrace{\texttt{if}\ \underbrace{x}_{t_x}\ \texttt{then}\ \underbrace{(x-1)}_{t_1}\ \texttt{else}\ \mathbf{0}}_{t_0}$$

$$\begin{array}{rcl} t_x & = & \text{bool} \\ t_1 & = & t_0 \\ \text{int} & = & t_0 \\ t_x & = & \text{int} \\ t_1 & = & \text{int} \end{array}$$

# Example 4

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ (\texttt{iszero}\ \underbrace{\underbrace{(f\ f)}_{t_2})}_{t_1}}_{t_0}$$

$$\begin{array}{rcl} t_0 & = & t_f \to t_1 \\ t_1 & = & \text{bool} \\ t_2 & = & \text{int} \\ t_f & = & t_f \to t_2 \end{array}$$

# 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.

- $$\frac{\Gamma \vdash E_1 : \mathsf{int} \qquad \Gamma \vdash E_2 : \mathsf{int}}{\Gamma \vdash E_1 + E_2 : \mathsf{int}}$$

$$t_{E_1} = \mathsf{int} \;\wedge\; t_{E_2} = \mathsf{int} \;\wedge\; t_{E_1+E_2} = \mathsf{int}$$

- $$\frac{\Gamma \vdash E : \mathsf{int}}{\Gamma \vdash \texttt{iszero}\ E : \mathsf{bool}}$$

$$t_E = \mathsf{int} \;\wedge\; t_{(\texttt{iszero}\ E)} = \mathsf{bool}$$

- $$\frac{\Gamma \vdash E_1 : t_1 \to t_2 \qquad \Gamma \vdash E_2 : t_1}{\Gamma \vdash E_1\ E_2 : t_2}$$

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

# Idea: Deriving Equations from Typing Rules

- $$\frac{\Gamma \vdash E_1 : \mathsf{bool} \qquad \Gamma \vdash E_2 : t \qquad \Gamma \vdash E_3 : t}{\Gamma \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 : t}$$

$$\begin{array}{rcl} t_{E_1} & = & \mathsf{bool} \wedge \\ t_{E_2} & = & t_{(\texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3)} \wedge \\ t_{E_3} & = & t_{(\texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3)} \end{array}$$

- $$\frac{[x \mapsto t_1]\Gamma \vdash E : t_2}{\Gamma \vdash \texttt{proc}\ x\ E : t_1 \to t_2}$$

$$t_{(\texttt{proc}\ (x)\ E)} = t_x \to t_E$$

- $$\frac{\Gamma \vdash E_1 : t_1 \qquad [x \mapsto t_1]\Gamma \vdash E_2 : t_2}{\Gamma \vdash \texttt{let}\ x = E_1\ \texttt{in}\ E_2 : t_2}$$

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

# 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.