ENE4014: Programming Languages

Lecture 11 — Type System (2) Design

Woosuk Lee

2024 Spring

Language

$$
\begin{array}{rcl}
E & \rightarrow & n \\
  & \mid & x \\
  & \mid & E + E \\
  & \mid & E - E \\
  & \mid & \texttt{iszero}\ E \\
  & \mid & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
  & \mid & \texttt{let}\ x = E\ \texttt{in}\ E \\
  & \mid & \texttt{proc}\ x\ E \\
  & \mid & E\ E
\end{array}
$$

Language

$$\frac{}{\rho \vdash n \Rightarrow n} \qquad \frac{}{\rho \vdash x \Rightarrow \rho(x)} \qquad \frac{\rho \vdash E_1 \Rightarrow n_1 \qquad \rho \vdash E_2 \Rightarrow n_2}{\rho \vdash E_1 + E_2 \Rightarrow n_1 + n_2}$$

$$\frac{\rho \vdash E \Rightarrow 0}{\rho \vdash \texttt{iszero}\ E \Rightarrow \mathit{true}} \qquad \frac{\rho \vdash E \Rightarrow n}{\rho \vdash \texttt{iszero}\ E \Rightarrow \mathit{false}}\ n \neq 0$$

$$\frac{\rho \vdash E_1 \Rightarrow \mathit{true} \qquad \rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v} \qquad \frac{\rho \vdash E_1 \Rightarrow \mathit{false} \qquad \rho \vdash E_3 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v}$$

$$\frac{\rho \vdash E_1 \Rightarrow v_1 \qquad [x \mapsto v_1]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{let}\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v}$$

$$\frac{}{\rho \vdash \texttt{proc}\ x\ E \Rightarrow (x, E, \rho)}$$

$$\frac{\rho \vdash E_1 \Rightarrow (x, E, \rho') \qquad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

Types

Types are defined inductively:

$$
\begin{array}{rcl}
T & \rightarrow & \mathsf{int} \\
  & | & \mathsf{bool} \\
  & | & T \rightarrow T
\end{array}
$$

Examples:

- int
- bool
- int $\rightarrow$ int
- bool $\rightarrow$ int
- int $\rightarrow$ (int $\rightarrow$ bool)
- (int $\rightarrow$ int) $\rightarrow$ (bool $\rightarrow$ bool)
- (int $\rightarrow$ int) $\rightarrow$ (bool $\rightarrow$ (bool $\rightarrow$ int))

Types of Expressions

In order to compute the type of an expression, we need *type environment*:

$$\boldsymbol{\Gamma : Var \to T}$$

Notation:

$\boldsymbol{\Gamma} \vdash \boldsymbol{e : t} \Leftrightarrow$ Under type environment $\boldsymbol{\Gamma}$, expression $\boldsymbol{e}$ has type $\boldsymbol{t}$.

Examples

- $[] \vdash \mathbf{3} : \mathsf{int}$
- $[x \mapsto \mathsf{int}] \vdash x : \mathsf{int}$
- $[] \vdash \mathbf{4} - \mathbf{3} :$
- $[x \mapsto \mathsf{int}] \vdash x - \mathbf{3} :$
- $[] \vdash \texttt{iszero}\ \mathbf{11} :$
- $[] \vdash \texttt{proc}\ (x)\ (x - \mathbf{11}) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{let}\ y = x - \mathbf{11}\ \texttt{in}\ (x - y)) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ \mathbf{11}\ \texttt{else}\ \mathbf{22}) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{proc}\ (y)\ \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ \mathbf{11}) :$
- $[] \vdash \texttt{proc}\ (f)\ (\texttt{if}\ (f\ \mathbf{3})\ \texttt{then}\ \mathbf{11}\ \texttt{else}\ \mathbf{22}) :$
- $[] \vdash (\texttt{proc}\ (x)\ x)\ \mathbf{1} :$
- $[f \mapsto \mathsf{int} \to \mathsf{int}] \vdash (f\ (f\ 1)) :$

Typing Rules

Inductive rules for assigning types to expressions:

$$\frac{}{\Gamma \vdash n : \mathsf{int}} \qquad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

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

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

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

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

We say that a closed expression E has type t iff we can derive $[] \vdash E : t$.

Example 1

$$\frac{}{[] \vdash \texttt{iszero}\ (\mathbf{1} + \mathbf{2}) : \mathsf{bool}}$$

Example 2

$$\frac{}{[] \vdash \texttt{proc}\ (x)\ (x - 11) : \mathsf{int} \rightarrow \mathsf{int}}$$

Example 3

$$\frac{}{[] \vdash \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ \mathbf{11}\ \texttt{else}\ \mathbf{22}) : \mathsf{bool} \to \mathsf{int}}$$

Example 4

$$\frac{}{[] \vdash (\texttt{proc}\ (x)\ x)\ 1 : \mathsf{int}}$$

Example 5

$$\frac{}{[] \vdash \texttt{proc}\ (x)\ (\texttt{proc}\ (y)\ \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ 11) : \mathsf{int} \to (\mathsf{bool} \to \mathsf{int})}$$

Property 1 (Multiple Types)

Type assignment may not be unique:

- $\texttt{proc}\ x\ x$:

$$\frac{[x \mapsto \mathsf{int}] \vdash x : \mathsf{int}}{[] \vdash \texttt{proc}\ x\ x : \mathsf{int} \to \mathsf{int}}$$

$$\frac{[x \mapsto \mathsf{bool}] \vdash x : \mathsf{bool}}{[] \vdash \texttt{proc}\ x\ x : \mathsf{bool} \to \mathsf{bool}}$$

$$\frac{[x \mapsto (\mathsf{int} \to \mathsf{int})] \vdash x : \mathsf{int} \to \mathsf{int}}{[] \vdash \texttt{proc}\ x\ x : (\mathsf{int} \to \mathsf{int}) \to (\mathsf{int} \to \mathsf{int})}$$

- $\texttt{proc}\ (f)\ (f\ 3)$ has type $(\mathsf{int} \to t) \to t$ for any t.
- The type of $\texttt{proc}\ (f)\ \texttt{proc}\ (x)\ (f\ (f\ x))$?

Property 2 (Soundness)

The type system is sound:

- If a closed expression $\boldsymbol{E}$ is well-typed

$$[] \vdash \boldsymbol{E} : \boldsymbol{t}$$

  for some $\boldsymbol{t} \in \boldsymbol{T}$, $\boldsymbol{E}$ does not have type error and produce a value:

$$[] \vdash \boldsymbol{E} \Rightarrow \boldsymbol{v}$$

- Furthermore, the type of $\boldsymbol{v}$ is $\boldsymbol{t}$. In other words, if $\boldsymbol{E}$ has a type error, we cannot find $\boldsymbol{t}$ such that $[] \vdash \boldsymbol{E} : \boldsymbol{t}$.
- Examples:
  - `(proc (x) x) 1`
  - `(proc (x) (x 3)) 4`

Property 2 (Soundness)

Theorem (Soundness of type system)

If E is a closed expression, $[] \vdash E : t \implies [] \vdash E \Rightarrow v$ *and* $v : t$

Proof.

- Case $E = n$: $[] \vdash E : \text{int}$, and $[] \vdash E \Rightarrow n$. Because $n : \text{int}$, the theorem holds.
- Case $E = x$: not a closed expression.
- Case $E = E_1 + E_2$: We will show $[] \vdash E : \text{int} \implies [] \vdash E \Rightarrow n$ for some integer n.
  Inductive hypothesis (IH): $[] \vdash E_1 : \text{int} \implies [] \vdash E_1 \Rightarrow n_1$ and $[] \vdash E_2 : \text{int} \implies [] \vdash E_2 \Rightarrow n_2$ for some integers n_1 and n_2.
  If $[] \vdash E_1 + E_2 : \text{int}$, then $[] \vdash E_1 : \text{int}$ and $[] \vdash E_2 : \text{int}$. By IH, $[] \vdash E_1 \Rightarrow n_1$ and $[] \vdash E_2 \Rightarrow n_2$. Therefore, $[] \vdash E_1 + E_2 \Rightarrow n_1 + n_2$ by the evaluation rules.
- The other cases: exercise

Property 3 (Incompleteness)

The type system is incomplete: even though some programs do not have type errors, they do not have types according to the type system:

- $\texttt{if iszero } \mathbf{1} \texttt{ then } \mathbf{11} \texttt{ else } (\texttt{iszero } \mathbf{22}))$
- $(\texttt{proc } (f)\ (f\ f))\ (\texttt{proc } x\ x)$

Property 3 (Incompleteness)

Why two branches should have the same type?

$$(\texttt{if}\ \boldsymbol{E}\ \texttt{then}\ \mathbf{1}\ \texttt{else}\ (\boldsymbol{true})) + \mathbf{1}$$

Can we know the value of $\boldsymbol{E}$ in advance?

- Case 1) $\boldsymbol{E}$ always evaluates to $\boldsymbol{true}$, so the program is OK!
- Case 2) $\boldsymbol{E}$ always evaluates to $\boldsymbol{false}$, so NOT OK!
- Case 3) $\boldsymbol{E}$ evaluate to $\boldsymbol{true}$ or $\boldsymbol{false}$, so NOT OK!
- Case 4) $\boldsymbol{E}$ never terminates, so OK!

We cannnot know the evaluation result of $\boldsymbol{E}$ without execution. If it's possible, we can solve the halting problem! (Why?)

Getting Results without Execution is Impossible

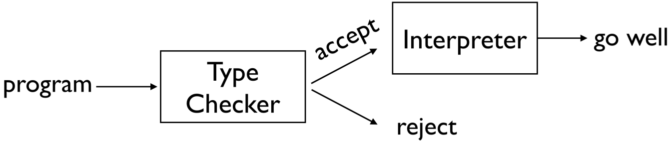
- Suppose we have an algorithm V that can exactly compute all possible values of a given expression p without execution.
- Then, we can construct $H(p)$ as follows:
  1. H takes p and construct the following program:

     $$p; \ \textit{true}$$

  2. Invoke the procedure V with this program
     - If $V = \{true\}$, then H returns yes (p terminates).
     - If $V = \emptyset$, then H returns no (p does not terminates).

Implementation

Implement a type checker according to the design:



- The type checker accepts a program $\boldsymbol{E}$ only if $[] \vdash \boldsymbol{E} : \boldsymbol{t}$ for some $\boldsymbol{t}$.
- Otherwise, $\boldsymbol{E}$ is rejected.