

# COSE212: Programming Languages

## Lecture 11 — Type System (2) Design

Hakjoo Oh  
2018 Fall

# Language

$$\begin{array}{l} E \rightarrow n \\ | \\ | x \\ | \\ | E + E \\ | \\ | E - E \\ | \\ | \text{iszero } E \\ | \\ | \text{if } E \text{ then } E \text{ else } E \\ | \\ | \text{let } x = E \text{ in } E \\ | \\ | \text{proc } x E \\ | \\ | E E \end{array}$$

# Language

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

$$\frac{\rho \vdash E \Rightarrow 0}{\rho \vdash \text{iszero } E \Rightarrow \text{true}} \quad \frac{\rho \vdash E \Rightarrow n}{\rho \vdash \text{iszero } E \Rightarrow \text{false}} \quad n \neq 0$$

$$\frac{\rho \vdash E_1 \Rightarrow \text{true} \quad \rho \vdash E_2 \Rightarrow v}{\rho \vdash \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \Rightarrow v} \quad \frac{\rho \vdash E_1 \Rightarrow \text{false} \quad \rho \vdash E_3 \Rightarrow v}{\rho \vdash \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \Rightarrow v}$$

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

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

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

# Types

Types are defined inductively:

$$\begin{array}{l} T \rightarrow \text{int} \\ | \text{bool} \\ | T \rightarrow T \end{array}$$

Examples:

- int
- bool
- $\text{int} \rightarrow \text{int}$
- $\text{bool} \rightarrow \text{int}$
- $\text{int} \rightarrow (\text{int} \rightarrow \text{bool})$
- $(\text{int} \rightarrow \text{int}) \rightarrow (\text{bool} \rightarrow \text{bool})$
- $(\text{int} \rightarrow \text{int}) \rightarrow (\text{bool} \rightarrow (\text{bool} \rightarrow \text{int}))$

## Types of Expressions

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

$$\Gamma : \mathit{Var} \rightarrow \mathit{T}$$

Notation:

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

## Examples

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

# Typing Rules

Inductive rules for assigning types to expressions:

$$\frac{}{\Gamma \vdash n : \text{int}} \quad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

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

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

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

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

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

# Example 1

$$\overline{\square \vdash \text{iszero } (1 + 2) : \text{bool}}$$



## Example 2

$$\overline{\square \vdash \text{proc } (x) (x - 11) : \text{int} \rightarrow \text{int}}$$

## Example 3

---

$\square \vdash \text{proc } (x) \text{ (if } x \text{ then } \mathbf{11} \text{ else } \mathbf{22}) : \text{bool} \rightarrow \text{int}$

## Example 4

$$\frac{}{\square \vdash (\text{proc } (x) x) 1 : \text{int}}$$

## Example 5

---

$$\boxed{\phantom{x}} \vdash \text{proc } (x) (\text{proc } (y) \text{ if } y \text{ then } x \text{ else } 11) : \text{int} \rightarrow (\text{bool} \rightarrow \text{int})$$

## Property 1 (Multiple Types)

Type assignment may not be unique:

- `proc x x:`

$$\frac{[x \mapsto \text{int}] \vdash x : \text{int}}{[] \vdash \text{proc } x \ x : \text{int} \rightarrow \text{int}}$$

$$\frac{[x \mapsto \text{bool}] \vdash x : \text{bool}}{[] \vdash \text{proc } x \ x : \text{bool} \rightarrow \text{bool}}$$

$$\frac{[x \mapsto (\text{int} \rightarrow \text{int})] \vdash x : \text{int} \rightarrow \text{int}}{[] \vdash \text{proc } x \ x : (\text{int} \rightarrow \text{int}) \rightarrow (\text{int} \rightarrow \text{int})}$$

- `proc (f) (f 3)` has type  $(\text{int} \rightarrow t) \rightarrow t$  for any  $t$ .
- The type of `proc (f) proc (x) (f (f x))?`

## Property 2 (Soundness)

The type system is sound:

- If a closed expression  $E$  is well-typed

$$\square \vdash E : t$$

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

$$\square \vdash E \Rightarrow v$$

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

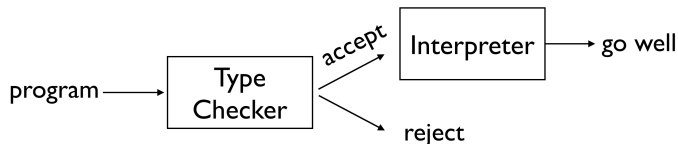
## Property 3 (Incompleteness)

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

- `if iszero 1 then 11 else (iszero 22))`
- `(proc (f) (f f)) (proc x x)`

# Implementation

Implement a type checker according to the design:



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