Lecture 21. Type Systems

Two-layered Design — Concurrent

Global layer $G_0 \to G_1 \to G_2 \to \cdots$

Local layer $e \overset{S}{\Rightarrow} v$ — $S, G \in$ Global States

$$\frac{\langle a, H_a \rangle \in G \quad [a \leftarrow v_m] \in G \quad H_a\ v_m \overset{S}{\Rightarrow} H_a'}{G \to G \setminus \{\langle a, H_a \rangle, [a \leftarrow v_m]\} \cup S \cup \langle a, H_a' \rangle}$$

S: new side effects; H_a': continuation of actor; H_a: old handler; H_a': new handler

$$\frac{e_1 \overset{S_1}{\Rightarrow} v_1 \quad e_2 \overset{S_2}{\Rightarrow} v_2 \quad v_1\ a\ v_2 \overset{S_3}{\Longrightarrow} v_h \quad a \text{ is an actor name not used before}}{\text{Create}(e_1, e_2) \overset{S_1 \cup S_2 \cup S_3 \cup \langle a, v_h \rangle}{\Longrightarrow} a}$$

S_1: actor constructor; S_2: init data; S_3

e_1: behavior; e_2: init data

actor = Fun me → Fun init_data → Fun msg -> ...

Quiz

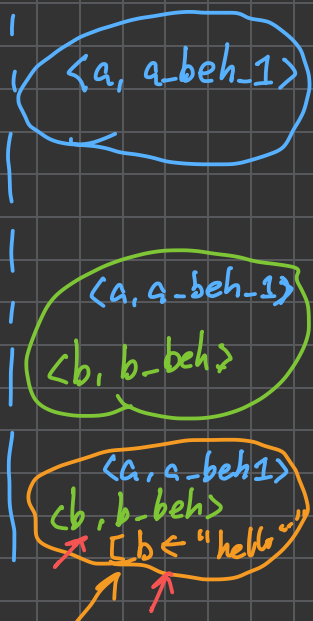
Let a_behavior = ...

Let a = Create (a_behavior. _) In

Let b_behavior = ...

Let b = Create (b_behavior. _) In

b ← "hello world"



$G_1 = \{ \langle a, a\_beh\_1 \rangle, \langle b, b\_beh \rangle, [b \leftarrow \text{"hello"}] \}_{G_1}$

two actors

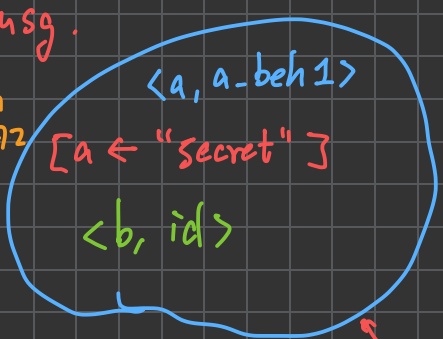
1 msg.

b_behavior : Fun msg →

(Print msg);

(a ← "the secret");

Fun x → x ID



$G_2 = \{ \langle a, a\_beh\_1 \rangle , [a \leftarrow \text{"secret"}], \langle b, id \rangle \}$

a_beh_1 = Fun msg

Print msg ← "The secret"

Ime ← " is 42 \n "):←

a_beh_2

$G_3 =$ <a, a_beh_2>

[a ← " is 42 "]

<b, id>

Hello world
Secret
is 42

$G_4 = \{$ $\}$

Lecture 21 Type Systems

Java C/C++ Rust Python JS

F^b Assembly ByteCode LISP ...

Statically Typed Langs: Java C/C++ Rust

Dynamically Typed Langs: (Java) Python JS LISP

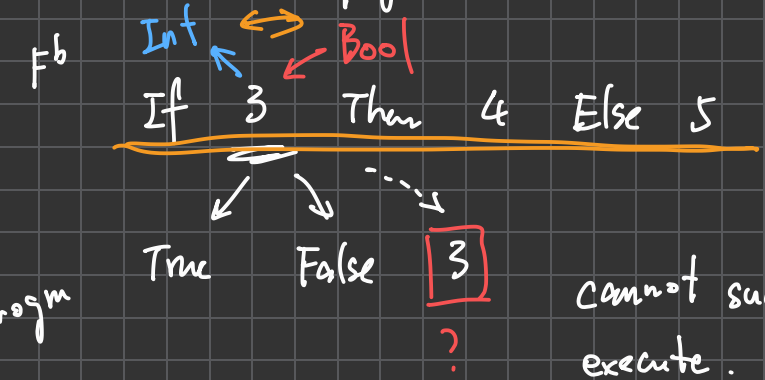
Untyped Langs: F^b Assm Bytecode

Py
```
i = 0
j = "hello"
i+j
```

F^b
```
let i = 0 In
let j = True In
i+j
```

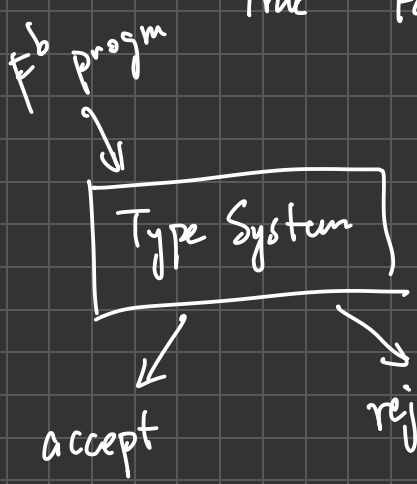
1. What does type offer

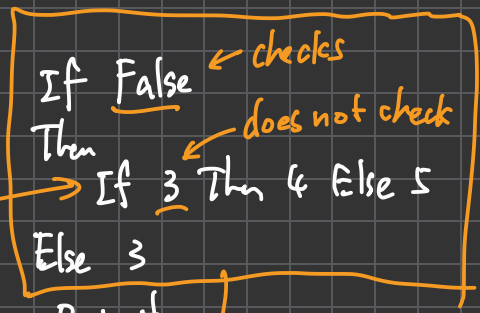
a. Tell whether a program can successfully execute

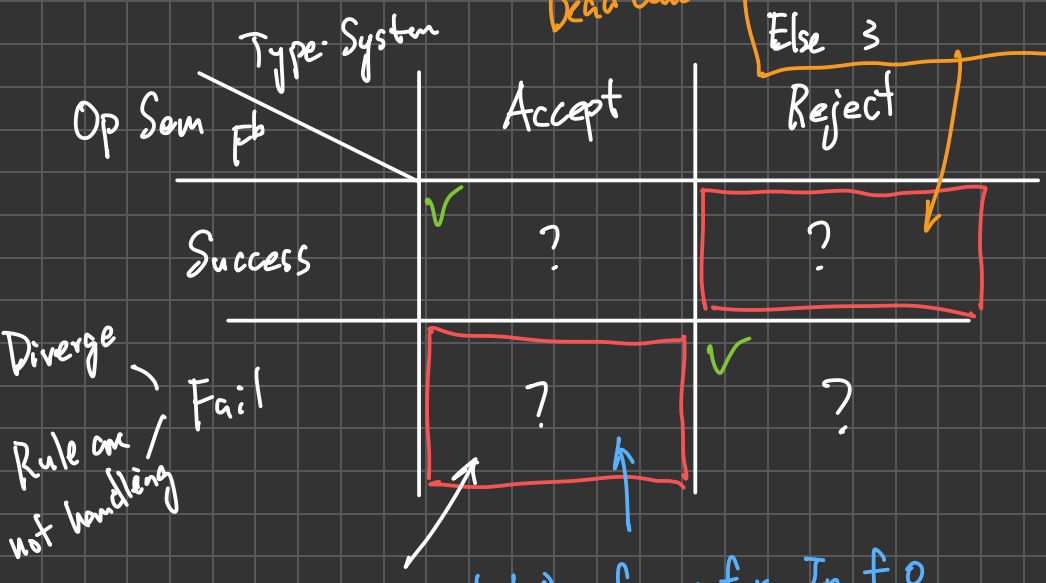


cannot successfully execute.

Reject







Let Rec f x = f x In f 0

What we expect from type system

Additional

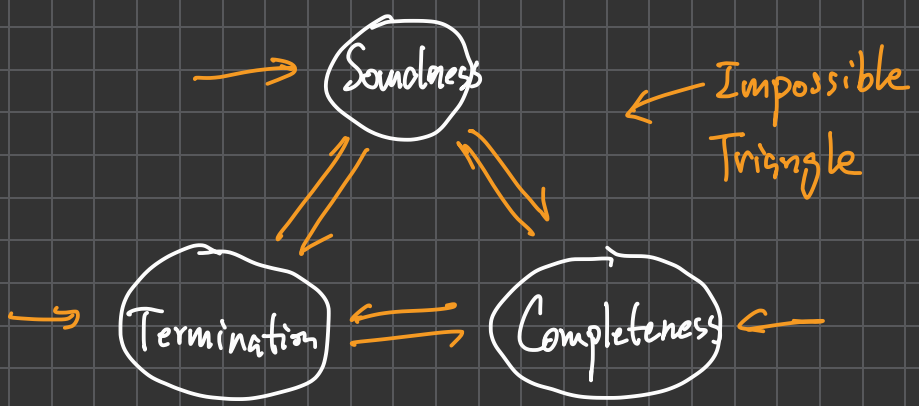
GT	Sys 1 \ Sys 2	T	F
	T	TP	False
	F	False Pos	TN

A system never gives False Positive : Soundness

A system never gives False Negative : Completeness

Type system accepts all programs that can run

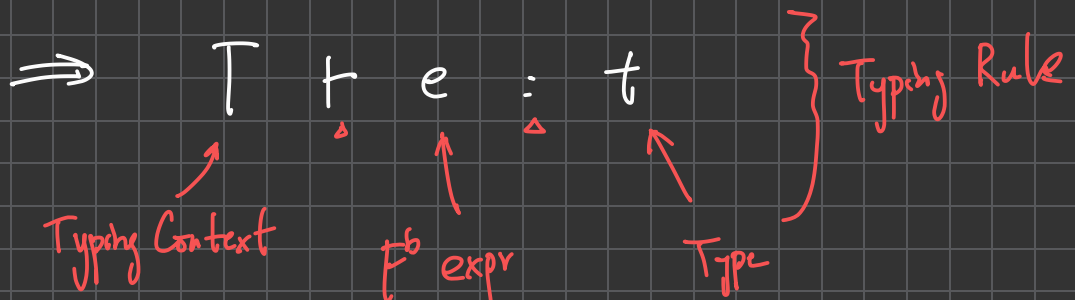
Type System ∈ Program Analyzers

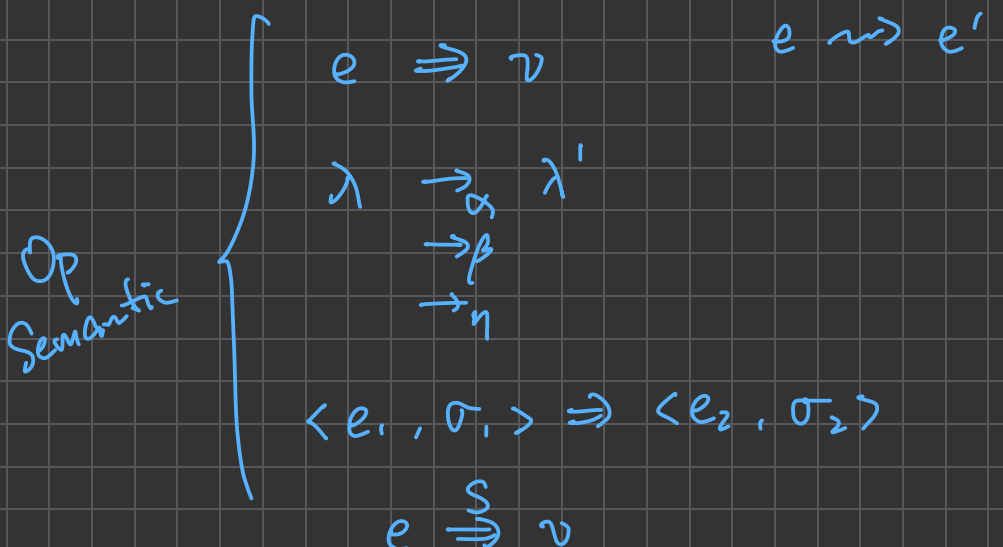


b. Optimization (Time - Space)

c. Ergonomics

Py. from typing import *





F^b e TF^b e t Γ ⊢ e : t