

Ocaml

KOSMOS

Small Ocaml Syntax

$$\begin{aligned} F &\rightarrow E \\ E &\rightarrow n \\ &| x \\ &| true \\ &| false \\ &| E + E \\ &| E - E \\ &| E * E \\ &| E / E \\ &| iszero E \\ &| if E then E else E \\ &| let [rec] x = E in E \\ &| fun x \rightarrow E \\ &| E E \\ &| read \end{aligned}$$

Small Ocaml Typing Rule

$$\frac{}{\Gamma \vdash n : \text{int}} \quad \frac{}{\Gamma \vdash x : \Gamma(x)} \quad \frac{}{\Gamma \vdash \text{true} : \text{bool}} \quad \frac{}{\Gamma \vdash \text{false} : \text{bool}} \quad \frac{}{\Gamma \vdash \text{read} : \text{int}}$$

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

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

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

$$\frac{[x \mapsto t_1] \Gamma \vdash E : t_2}{\Gamma \vdash \text{fun } x \rightarrow E : t_1 \rightarrow t_2} \quad \frac{\Gamma \vdash E_1 : t_1 \rightarrow t_2 \quad \Gamma \vdash E_2 : t_1}{\Gamma \vdash E_1 E_2 : t_2}$$

Too Sound Type Checking

```
let id = fun x -> x in  
let x = read in  
if id (iszero x) then id (x + 1) else id (x - 1)
```

=> Rejected.

Too Sound Type Checking

```
let id = fun x -> x in  
let x = read in  
if id (iszero x) then id (x + 1) else id (x - 1)
```

=> Rejected.

In Ocaml,

```
> 1  
2
```

```
> 0  
-1
```

Polymorphic Type System

하나의 함수 프로그램이 서로 다른 타입 환경에서 실행가능한 언어

```
let id = fun x -> x in  
let x = read in  
if id (iszero x) then id (x + 1) else id (x - 1)
```

$$id(iszero\ x) = bool \rightarrow bool$$

Polymorphic Type System

하나의 함수 프로그램이 서로 다른 타입 환경에서 실행가능한 언어

```
let id = fun x -> x in  
let x = read in  
if id (iszero x) then id (x + 1) else id (x - 1)
```

$$\begin{aligned} id(x + 1) &= int \rightarrow int \\ id(x - 1) &= int \rightarrow int \end{aligned}$$

One Solution

```
let x = fun x -> x in  
id (iszero (id 1))
```


One Solution

```
let x = fun x -> x in  
id (iszero (id 1))
```

```
(fun x -> x) (iszero ((fun x -> x) 1))
```

How to Implement

- Environment와 비슷한 ExpEnv를 이용

$$\text{ExpEnv} = \text{Var} \rightarrow \text{Exp}$$

- LET 에서 모두 바꿈

$$\text{replace}: \text{exp} \rightarrow \text{var} \rightarrow \text{exp} \rightarrow \text{exp}$$

Problem of This Solution

let x = 1 + true in 3

```
let x = 1 + true in 3
```

실제 Type Error가 있는 프로그램이지만 검사할 수 없음

Simplie Solution

let x = 1 + true in 3

let x = 1 + true in 3

Simple Solution

- 변수가 이후에 사용되는지 미리 확인

check_occurence: exp → var → bool

- 바꾼 후 실제로 바뀌었는지 확인

*updated_replace: exp → var → exp → exp * bool*