Ocaml

KOSMOS

Small Ocaml Syntax

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

Small Ocaml Typing Rule

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

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

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

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

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

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)
```

$$id(x+1) = int \rightarrow int$$
$$id(x-1) = int \rightarrow int$$

One Solution

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

One Solution

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

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

How to Implement

- Environment와 비슷한 ExpEnv를 이용

$$ExpEnv = Var \rightarrow Exp$$

- LET 에서 모두 바꿈

$$replace: exp \rightarrow var \rightarrow exp \rightarrow exp$$

Problem of This Solution

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

Problem of This Solution

```
let x = 1 + true in 3                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

$$check\_occurence : exp \rightarrow var \rightarrow bool$$

- 바꾼 후 실제로 바뀌었는지 확인

$$updated\_replace : exp \rightarrow var \rightarrow exp \rightarrow exp * bool$$