VI. Type Systems

To learn:
- pl 에서의 type 의 역할
- simple type system
- polymorphic type system
- type checking / type inference
- existential type systems
- subtype system
- typed compilation

* key ideas about the role of types in p.l.
* mathematical formalism of type systems

Static Semantics

Typing Formula ,judgment, 판단

$$\boxed{\Gamma \vdash e : \tau}$$

antecedent consequent (predicate)

“식 e가 타입에 맞게 계산이 되고 결과의 타입은 tau 이다.”

$\Gamma \in \text{Vars} \xrightarrow{\text{fin}} \text{Types}$

type assignment

type environment

$e \in \text{Expressions}$

$\tau \in \text{Types}$

$(\Gamma + x:\tau \equiv \Gamma[\tau/x]$ where $\text{dom}(\Gamma[\tau/x])$ is expanded if necessary$)$

The Language

Syntax

$$e \to x \mid () \mid \lambda x.e \mid e\ e$$
$$\mid \text{let } x = e \text{ in } e$$

Types

$$\tau \to \iota \mid \tau \to \tau$$

* Semantics

· $e \xrightarrow{*} v$ transitive closure

Logical rules, not algorithmic definitions

Static Semantics

(VAR)
$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

(FN)
$$\frac{\Gamma + x:\tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2}$$

(APP)
$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$$

(LET)
$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma + x:\tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : \tau_2}$$

(CONST)
$$\Gamma \vdash () : \iota$$

Static Semantics

Note 1 $\exists 1$ typing rule for each expression construct.

Note 2 Some expression can have mutiple types.

$$\lambda x.x : \iota \to \iota$$
$$(\iota \to \iota) \to (\iota \to \iota)$$
$$\vdots$$

Note 3 let $I = \lambda x.x$ in II cannot have a type.

Correctness of the Type System

Theorem [Type Safety]

Lemma [Progress]
Lemma [Preservation]

자 이제,
타입시스템을
구현하는 방안을
강구하자.

기본은 이거다:
프로그램을 스캔해서
풀어야할 방정식을
도출하고, 그 방정식의
해를 계산한다.

그럼, 확인할게 있다.
그런 방식이 맞는가?

Type Checking

- Determine for given Γ, e, and τ whether $\Gamma \vdash e : \tau$ or not.

$\Longrightarrow$ (reduce) Check provability in a simple logical system called unification logic.

- U.L. Formula

$$\varphi ::= \tau_1 \doteq \tau_2 \mid \varphi \wedge \varphi \mid \exists \alpha . \varphi$$

- U.L. Proof Rules

$$\vdash \tau \doteq \tau \qquad \frac{\vdash \varphi_1 \quad \vdash \varphi_2}{\vdash \varphi_1 \wedge \varphi_2}$$

$$\frac{\vdash [\tau/\alpha]\varphi}{\vdash \exists \alpha . \varphi}$$

Type Checking $\xRightarrow{\text{reduce}}$ Provability in U.L.

Define a constraint generation algorithm

$$\nabla(\Gamma, e, \tau) = \varphi$$

s.t. $\vdash \varphi$ iff $\Gamma \vdash e : \tau$

프로그램을
스캔해서
방정식
만들어내는
방법 V

방정식 $V(\Gamma, e, \tau)$ 의 해가 있으면

$\Gamma \vdash e : \tau$ 이 사실이 되도록.

근데, 반대도 성립할 필요가 있을까?

프로그램을 스캔해서 방정식 만들어내는 방법 V

Type Checking $\underset{\text{reduce}}{\Rightarrow}$ Provability in U.L.

Define a constraint generation algorithm

$$\nabla(\Gamma, e, \tau) = \varphi$$

s.t. $\vdash \varphi$ iff $\Gamma \vdash e : \tau$

Let

$$\nabla(\Gamma, x, \tau) = \Gamma(x) \doteq \tau$$

$$\nabla(\Gamma, \lambda x.e, \tau) =$$
$$\exists \alpha_1, \alpha_2.\ \tau \doteq \alpha_1 \rightarrow \alpha_2 \ \wedge \ \nabla(\Gamma + x:\alpha_1, e, \alpha_2)$$

$$\nabla(\Gamma, e_1 e_2, \tau) =$$
$$\exists \alpha.\ \nabla(\Gamma, e_1, \alpha \rightarrow \tau) \wedge \nabla(\Gamma, e_2, \alpha)$$

$$\nabla(\Gamma, \text{let } x = e_1 \text{ in } e_2, \tau) =$$
$$\exists \alpha.\ \nabla(\Gamma, e_1, \alpha) \wedge \nabla(\Gamma + x:\alpha, e_2, \tau)$$

∇ Example

$$\nabla(\phi, (\lambda x.x)(\lambda x.x), \tau) =$$

$$\exists \alpha_1.\ \nabla(\phi, \lambda x.x, \alpha_1 \to \tau) \wedge \nabla(\phi, \lambda x.x, \alpha_1)$$

$$= \exists \alpha_1. (\exists \alpha_2, \alpha_3.\ \nabla(x:\alpha_2, x, \alpha_3) \wedge \alpha_2 \to \alpha_3 \doteq \alpha_1 \to \tau$$
$$\exists \alpha_4, \alpha_5.\ \nabla(x:\alpha_4, x, \alpha_5) \wedge \alpha_1 \doteq \alpha_4 \to \alpha_5)$$

$$= \exists \alpha_1 (\exists \alpha_2, \alpha_3.\ \alpha_2 \doteq \alpha_3 \wedge \alpha_2 \to \alpha_3 \doteq \alpha_1 \to \tau$$
$$\exists \alpha_4, \alpha_5.\ \alpha_4 \doteq \alpha_5 \wedge \alpha_1 \doteq \alpha_4 \to \alpha_5)$$

$$= \exists \alpha_1 \alpha_2 \alpha_3 \alpha_4 \alpha_5. (\alpha_2 \doteq \alpha_3 \wedge \alpha_2 \to \alpha_3 \doteq \alpha_1 \to \tau$$
$$\wedge\ \alpha_4 \doteq \alpha_5 \wedge \alpha_1 \doteq \alpha_4 \to \alpha_5)$$

A system of equations to be solved.

```
a1 = a2
a3 = tau
a2 = tau
a1 = a4 -> a4
```

```
a1 = a2 = a3 = tau
a1 = a4 -> a4
a4 = a5
```

V({}, \x.xx, t)

t = a1->a2

V({x:a1}, xx, a2)

V({x:a1}, x, a3->a2)   V({x:a1}, x, a3)

a1 = a3->a2   a1 = a3

a1 = a1->a2

위의 방정식을 만족시킬 a1 이 있는가?

방정식의 해를 어디서 찾아야하나요? (Z, N, Q, R?)

{I, I->I, I->I->I, (I->I)->I, ...}

즉, t = I | t->t

그렇게만들어낸 방정식의 해
=
타입시스템에서 유추하는 타입?

computation
=
logic?

implementation
=
design?

Then

$\vdash \varphi$ iff $\Gamma \vdash e : \tau$

where $\nabla(\Gamma, e, \tau) = \varphi$

pr) S.I. on e

case $e \equiv e_1 e_2$

(⇒) $\vdash \varphi = \nabla(\Gamma, e_1 e_2, \tau)$

$= \exists \alpha. \nabla(\Gamma, e_1, \alpha \to \tau) \wedge \nabla(\Gamma, e_2, \alpha)$

then $\vdash \nabla(\Gamma, e_1, \tau_2 \to \tau) \wedge \nabla(\Gamma, e_2, \tau_2)$

for some τ_2

then $\vdash \nabla(\Gamma, e_1, \tau_2 \to \tau) \overset{S.I.}{\iff} \Gamma \vdash e_1 : \tau_2 \to \tau$

$\vdash \nabla(\Gamma, e_2, \tau_2) \iff \Gamma \vdash e_2 : \tau_2$

∴ $\Gamma \vdash e_1 e_2 : \tau$

(⇐) $\Gamma \vdash e_1 e_2 : \tau$

then $\Gamma \vdash e_1 : \tau_2 \to \tau \overset{S.I.}{\iff} \vdash \nabla(\Gamma, e_1, \tau_2 \to \tau)$

$\Gamma \vdash e_2 : \tau_2 \iff \vdash \nabla(\Gamma, e_2, \tau_2)$

then $\vdash \exists \alpha. \nabla(\Gamma, e_1, \alpha \to \tau) \wedge \nabla(\Gamma, e_2, \alpha)$

$= \nabla(\Gamma, e_1 e_2, \tau)$

Thm Let $\nabla(\Gamma, e, \tau) = \exists \alpha_1, \cdots, \alpha_n . \overline{\nabla(\Gamma, e, \tau)}$.

Either

① not $\vdash \nabla(\Gamma, e, \tau)$ or

② $\exists 1$ substitution $S \in \{\alpha_1, \cdots, \alpha_n\} \to \text{Types}$

such that $\vdash S(\overline{\nabla(\Gamma, e, \tau)})$.

pr) corollary of the Unification Theorem and the Resolution Principle.

"A Machine-Oriented Logic Based on the Resolution Principle"
Robinson, JACM Vol. 12, No. 1, pp. 23-41, 1965.

Such system of equations $\nabla(\Gamma, e, \tau)$ always has, if any, a unique solution.

Notation

$\overline{\nabla(\Gamma, e, \tau)}$ 는 $\nabla(\Gamma, e, \tau)$ 에서 $\exists \alpha_1, \cdots, \alpha_n .$ 을 제거한 것.

```
type ty = I | Fn of ty*ty | V of string
exception UFail

fun unify(t,t') = if t=t' then {} else
   case (t, t')
   of (Fn(t1,t2), Fn(t1',t2'))
      => let
            val s1 = unify(t1,t1')
            val s2 = unify(s1 t2, s1 t2')
         in
            s2 s1
         end

   | (V a, t) => {t/a} when a   t        ∉

   | (t, V a) => {t/a} when a   t        ∉

   | _ => raise UFail
```

Type Inference

Determine, for given Γ and e the τ such that $\Gamma \vdash e : \tau$.

Use the same procedure $\nabla(\Gamma, e, \alpha)$ but with a type var α.

Thm Let $\nabla(\Gamma, e, \alpha) = \exists \alpha_1, \cdots, \alpha_n . \overline{\nabla(\Gamma, e, \alpha)}$

Either

① not $\vdash \exists \alpha . \nabla(\Gamma, e, \alpha)$ or

② $\exists$ principal substitution

$S \in \{\alpha, \alpha_1, \cdots, \alpha_n\} \to$ TypesWithVars

such that

$\vdash S(\overline{\nabla(\Gamma, e, \alpha)})$

- Principal S

  if $\vdash S'(\overline{\nabla(\Gamma, e, \alpha)})$ then $\exists T$ s.t. $S' = TS$.

- TypesWithVars $\tau \to \iota \mid \tau \to \tau \mid \alpha$

  type variable

pr) Robinson's Unification Theorem & Resolution Principle

tcoML 을 roverML로!

문제 I 현재의 안전한 타입시스템을 통과하는 재귀함수는 없다.

```
let
  fac = Y λf. λn. if n=0 then 1 else n*f(n-1)
in
  fac 28
```

은 타입 시스템이 거부함.

$\because\ Y \equiv \lambda F.\lambda x. \cdots \underbrace{xx}\cdots$

self application "xx" has no type.

해결 ① 재귀함수의 정의가 설탕이 아니도록:

$$e \to \vdots \;\;|\; \mathrm{rec}\ f\ \lambda x.e$$

② 실행 의미는 정의 : "우리다 알아요!" $\quad v \to \vdots \;\;|\; \mathrm{rec}\ f\ \lambda x.e$

③ 타입 시스템 정의:

$$(\mathrm{REC})\ \frac{\Gamma + f:\tau \vdash \lambda x.e : \tau}{\Gamma \vdash \mathrm{rec}\ f\ \lambda x.e : \tau}$$

④ 타입 시스템이 안전한지 확인 : 증명해야죠.

⑤ 안전한 타입 시스템의 충실한 구현이 맞는지 확인 : 증명.

문제 II 메모리 반응/주소를 표현하기가?

해결 ① 주소값을 만드는 방법과 사용하는 방법을 제공하자:

$$
\begin{aligned}
e \rightarrow\ & \vdots \\
& \mid \ \mathtt{malloc}\ e \\
& \mid \ e := e \\
& \mid \ !e
\end{aligned}
$$

② 실행 의미는 정의 : "다 알아요!" ($\sigma, M \vdash e \Rightarrow v, M'$)

$$\boxed{\langle e, M\rangle \rightarrow \langle e', M'\rangle}$$

$$\langle \mathtt{malloc}\ v, M\rangle \rightarrow \langle \ell, M[v/\ell]\rangle \qquad \ell \notin \mathrm{dom}\, M$$

$$\langle \ell := v, M\rangle \rightarrow \langle v, M[v/\ell]\rangle \qquad \ell \in \mathrm{dom}\, M$$

$$\langle !\ell, M\rangle \rightarrow \langle M(\ell), M\rangle \qquad \ell \in \mathrm{dom}\, M$$

$$\frac{\langle e, M\rangle \rightarrow \langle e', M'\rangle}{\langle E[e], M\rangle \rightarrow \langle E[e'], M'\rangle}$$

$$
\begin{aligned}
E \rightarrow\ & \vdots \\
& \mid \ \mathtt{malloc}\ E \\
& \mid \ E := e \\
& \mid \ \ell := E \\
& \mid \ !E
\end{aligned}
$$

$$
\begin{aligned}
v \rightarrow\ & \vdots \\
& \mid \ \ell
\end{aligned}
$$

③ 타입 시스템 정의 : $\tau \to \iota \mid \tau \to \tau \mid \tau\ \mathrm{loc}$

$$(\mathrm{MALLOC})\quad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathrm{malloc}\ e : \tau\ \mathrm{loc}}$$

$$(\mathrm{ASS})\quad \frac{\Gamma \vdash e_1 : \tau\ \mathrm{loc} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 := e_2 : \tau}$$

$$(\mathrm{ACC})\quad \frac{\Gamma \vdash e : \tau\ \mathrm{loc}}{\Gamma \vdash\ !e : \tau}$$

④ 타입 시스템이 안전한지 확인 : 증명.

⑤ 안전한 타입 시스템의 충실한 구현이 맞는지 확인 : 증명.

$$\nabla(\Gamma, \mathrm{malloc}\ e, \tau) = \exists \alpha.\ \tau \doteq \alpha\ \mathrm{loc} \wedge \nabla(\Gamma, e, \alpha)$$

$$\nabla(\Gamma, e_1 := e_2, \tau) = \nabla(\Gamma, e_1, \tau\ \mathrm{loc}) \wedge \nabla(\Gamma, e_2, \tau)$$

$$\nabla(\Gamma, !e, \tau) = \nabla(\Gamma, e, \tau\ \mathrm{loc})$$