

$PT(\lambda x.P)$ is easy.

under $PT(PQ)$.

let $PT(P) \equiv P \rightarrow \sigma$ $PT(Q) \equiv \tau$

let $PT(PQ)$ has no free variables,

we need to make P and τ same in order to apply $(\rightarrow E)$.

let S_1, S_2 be substitutions s.t. $\frac{P \vdash S_1(P) \rightarrow S_1(\sigma) \quad P \vdash S_2(\tau)}{P \vdash PQ : S_1(\sigma) (\rightarrow E)}$

$$S_1(P) \equiv S_2(\tau)$$

Then $PT(PQ) \equiv S_1(\sigma)$ by $(\rightarrow E)$

Thus the problem of deciding whether PQ is typable reduces to that of finding S_1 and S_2 s.t. $S_1(P) \equiv S_2(\tau)$.

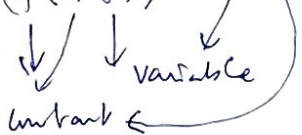
Definition (Common instance (c.i.))

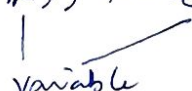
†† If $V \equiv S_1(P) \equiv S_2(\tau)$ we call V an c.i. of the pair (P, τ) and we call (S_1, S_2) a pair of unifying substitutions for (P, τ) .

††
eg.

~~$P \equiv (a, b \rightarrow b)$~~ ~~$\tau \equiv (f(1, y), f(x, 2))$~~

1. $(a, b \rightarrow b)$ $S_1 \equiv [b \rightarrow b/a]$ $S_2 \equiv [b/b]$

2. $(f(1, y), f(x, 2))$ $S_1 \equiv [2/y]$ $S_2 \equiv [1/x]$


3. $(foo(a, x), foo(y, b))$ $S_1 \equiv [b/x]$, $S_2 \equiv [a/y]$


4. $(foo(a, x, c), foo(y, 2))$ no substitution exists.
 for (P, τ)

Defn Most general c.i. (mgci) is a c.i. v_0

s.t. every other c.i. v is an instance of v_0 .

1. $S_2 \equiv \text{empty}$
2. mgci
3. mgci

The principal-type algorithm (PT-algorithm) (type checking algorithm).

$\text{id} = \lambda x. x$, $\text{id} : a \rightarrow a$ same as $\text{id} : b \rightarrow b$ *
 The type $(a \rightarrow a)$ is called the principal type of id (simplest form).
 No other type that can be assigned to id ~~has~~ is simpler than $(a \rightarrow a)$. For eg. $(a \rightarrow a) \rightarrow (a \rightarrow a)$. In fact all other types for id are substitution instances of the basic type $(a \rightarrow a)$.

Definition (Type-Substitution) :-

A type substitution **S** (bold face S) is any expression
 $[\sigma_1/a_1, \dots, \sigma_n/a_n] \leftarrow$ simultaneously substituting σ_i for a_i
 a_i 's: type variables - distinct - σ_i : any type

- (i) $S(a_i) \equiv \sigma_i$
- (ii) $S(b) \equiv b$ if b is an atom $\notin \{a_1, \dots, a_n\}$
- (iii) $S(p \rightarrow \tau) \equiv S(p) \rightarrow S(\tau)$

We call $S(\tau)$ an instance of τ . (substitution-instance of τ)

Definition (PT) A PT of a term M (in TA_λ) is a type τ s.t.

- (i) $\Gamma \vdash_\lambda M : \tau$ for some Γ
- (ii) if $\Gamma' \vdash_\lambda M : \sigma$ for some Γ' and σ then σ is an instance of τ .

Note: • A type τ is a PT of M iff,
 for all types σ , $(\exists \Gamma'. \Gamma' \vdash_\lambda M : \sigma) \Leftrightarrow \sigma$ is an instance of τ .
 • A PT of a term is unique. $\text{PT}(M)$ - denotes PT of M .

* alphabetic variants.