Motivating the PT algorithm

11.4.2025.

PT(λx.P) is easy.

Consider PT(PQ).

. let PT(P) ≡ P → σ PT(Q) ≡ τ

Let PT(PQ) has no free variables,

we need to make P and τ same in order to apply (→E)

let S_1, S_2 be substitutions s.t.

$$\frac{\Gamma \mapsto^{P:} S_1(P) \to S_1(\sigma) \quad \Gamma \mapsto^{Q:} S_2(\tau)}{\Gamma \mapsto PQ : S_1(\sigma)\ (\to E)}$$

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

Then PT(PQ) ≡ $S_1(\sigma)$ by (→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.))

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

(ii)

eg.

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

   variable, instance
3. (foo(a, X), foo(Y, b)) $S_1 \equiv [b/X]$, $S_2 \equiv [a/Y]$

   variable
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.

v = S(v0)

1. $S_2 \equiv$ empty 2. mgci 3. mgci

11-2-2025

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

$id = \lambda x.x$, $id : a \to a$ same as $id : b \to b$ *

The type $(a \to a)$ is called the principal type of id (simplest form).

No other type that can be assigned to id ~~less~~ is simpler than $(a \to a)$. For eg. $(a \to a) \to (a \to a)$, Infact all other types for id are substitution instances of the basic type $(a \to a)$.

Definition (Type-substitution) :-

A type substitution **S** (bold face S) is any expression

$[\sigma_1/a_1, \dots, \sigma_n/a_n]$ ← simultaneously substituting

a_i's: type variables – distinct – σ_i : any type σ_i for a_i

(i) $S(a_i) \equiv \sigma_i$

(ii) $S(b) \equiv b$ if b is an atom $\notin \{a_1 \dots a_n\}$

(iii) $S(\rho \to \tau) \equiv S(\rho) \to S(\tau)$

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, $(\exists \Gamma'.\ \Gamma' \vdash_\lambda M : \sigma) \iff \sigma$ is an instance of τ.

• A PT of a term is unique. PT(M) – denotes PT of M

---

* alphabetic variants.

means there is some type substitution which will lead to sigma from tau.