input = lambda term
output = principle deduction which will give principle type of input term.

21·4·2025

PT Algorithm   1969 — Hindley

Input : any λ-term M, closed ~~or not~~

Intended output : either a principal deduction Δ_M for M or a correct statement that M is not typable.

Case I : M is a variable $M \equiv x$

choose Δ_M to be the one-formula deduction

$x : a \mapsto x : a$ where a is any type variable.

(Δ_M is principal for M).

Case II : If $M \equiv \lambda x . P$ and $x \in FV(P)$, say $FV(P) = \{x, x_1, \ldots x_t\}$

apply the algorithm to P.

If P is not typable, neither is M.

If P has a principal deduction Δ_P its conclusion must be of the form

$$x : \alpha,\ x_1 : \alpha_1, \ldots,\ x_t : \alpha_t \ \mapsto\ P : \beta$$

for some types $\alpha, \alpha_1, \ldots, \beta$. Apply $(\to I)$main to obtain

$$x_1 : \alpha_1 \ldots\ x_t : \alpha_t \ \mapsto\ (\lambda x . P) : \alpha \to \beta$$

Call this deduction $\Delta_{\lambda x P}$

Case III : If $M \equiv \lambda x . P$ and $x \notin FV(P)$, say $FV(P) = \{x_1 \ldots x_t\}$

apply the algo. to P. If P is not typable, neither is M.

If P has a principal deduction Δ_P its conclusion must be of the form

$$x_1 : \alpha_1 \ldots\ x_t : \alpha_t \ \mapsto\ P : \beta$$

for some types $\alpha_1 \ldots, \beta$.

Choose a new type variable a not in Δ_P. and apply $(\to I)_{vac}$, vacuously discharging $x : a$ to get a deduction

$$x_1 : \alpha_1 \ldots\ x_t : \alpha_t \ \mapsto\ (\lambda x . P) : a \to \beta$$

Call this deduction $\Delta_{\lambda x P}$

Case IV. If $M \equiv PQ$, apply the algo. to P and Q. If P or Q is untypable then so is M. If P and Q are both typable, let Δ_P, Δ_Q be their principal deductions.

First rename type-variables, if necessary, to ensure that ~~no~~ Δ_P and Δ_Q have no common type-variables.

Next list the free term-variables in P and those in Q (these lists may overlap): say

$FV(P) = \{u_1 \dots u_p, w_1 \dots w_r\}$ $\quad p, r \geq 0$

$FV(Q) = \{v_1 \dots v_q, w_1 \dots w_r\}$ $\quad q \geq 0$

where $u_1 \dots u_p$, $v_1 \dots v_q$, $w_1 \dots w_r$ are distinct.

the type of P in case 4 is not atomic.

Subcase IV a. $M \equiv PQ$ and $PT(P) \equiv \rho \to \tau$.

[1] Δ_P : $u_1:\theta_1 \dots u_p:\theta_p, w_1:\varphi_1 \dots w_r:\varphi_r \mapsto P:\rho\to\sigma$

[2] Δ_Q : $v_1:\phi_1 \dots v_q:\phi_q, w_1:\chi_1 \dots w_r:\chi_r \mapsto Q:\tau$

Apply the unification algo. to the pair of sequences

[*] $\langle \varphi_1, \dots \varphi_r, \rho\rangle$, $\langle \chi_1 \dots \chi_r, \tau\rangle$.

IV.a.1 : [*] has no ~~unif~~ unifier. Then PQ is not typable.

IV.a.2 : [*] has a unique (m.g.u.) $\mathcal{U}$.

apply $\mathcal{U}$ to Δ_P, Δ_Q to obtain

$u_1:\theta_1^* \dots u_p:\theta_p^*, w_1:\varphi_1^* \dots w_r:\varphi_r^* \mapsto P:\rho^*\to\sigma^*$

$v_1:\phi_1^* \dots v_q:\phi_q^*, w_1:\chi_1^* \dots w_r:\chi_r^* \mapsto Q:\tau^*$

where $\theta_1^* = \mathcal{U}(\theta_1)$ etc.. By the definition of $\mathcal{U}$,

$\varphi_1^* = \chi_1^*$ etc, $\rho^* = \tau^*$ Now $(\to E)$ can be applied.

Call the resulting deduction Δ_{PQ}. where $PQ : \sigma$

Subcase IV b. $M \equiv PQ$, $PT(P) \equiv b$ (atomic)

let c be a type variable that does not occur in [1] and [2].

Apply the unif^n algo. to as in [*] where $\rho \equiv b$, $\tau \equiv \tau \to c$

IV.b.1 : the pair has no unifier. Then PQ is not typable.

IV.b.2. the pair has a unifier (mgu). ~~then~~

$\mathcal{U}(b) = \mathcal{U}(\tau \to c) \equiv \mathcal{U}(\tau) \to \mathcal{U}(c)$ Now $(\to E)$ can be applied

$PQ : c^*$ Use the above steps as in IV.a.2.

principle type algorithm will always produce principle type.