

# Formalization of Pure Type System

## 1. Definition

(i) A *pure type system (PTS)* is a triple tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{R})$  where

- (a)  $\mathcal{S}$  is a set of *sorts*;
- (b)  $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$  is a set of *axioms*;
- (c)  $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$  is a set of *rules*.

(ii) *Raw expressions*  $A$  and *raw environments*  $\Gamma$  are defined by

$$\begin{aligned} A &::= x \mid s \mid AA \mid \lambda x : A. A \mid \Pi x : A. A \\ \Gamma &::= \emptyset \mid \Gamma, x : A \end{aligned}$$

where we use  $s, t, u$ , etc. to range over sorts,  $x, y, z$ , etc. to range over variables, and  $A, B, C, a, b, c$ , etc. to range over expressions.

(iii)  $\Pi$  and  $\lambda$  are used to bind variables. Let  $\text{FV}(A)$  denote free variable set of  $A$ . Let  $A[x := B]$  denote the substitution of  $x$  in  $A$  with  $B$ . Standard notational conventions are applied here. Besides we also define  $A \rightarrow B$  as  $(\Pi x : A. B)$  where  $x \notin \text{FV}(B)$ .

(iv) The relation  $\rightarrow_\beta$  is the smallest binary relation on raw expressions satisfying

$$(\lambda x : A. M)N \rightarrow_\beta M[x := N]$$

which can be used to define the notation  $\twoheadrightarrow_\beta$  and  $=_\beta$  by convention.

(v) Type assignment rules for  $(\mathcal{S}, \mathcal{A}, \mathcal{R})$  are given in Table 1.

## 2. Examples of PTSs

(i) The  $\lambda$ -cube consists of eight PTSs, where

- (a)  $\mathcal{S} = \{\star, \square\}$
- (b)  $\mathcal{A} = \{(\star, \square)\}$
- (c)  $\{(\star, \star)\} \subseteq \mathcal{R} \subseteq \{(\star, \star), (\star, \square), (\square, \star), (\square, \square)\}$

Note that here we slightly abuse the notation of the set of rules  $\mathcal{R}$ , since in PTSs,  $\mathcal{R}$  is a ternary relation, while in the  $\lambda$ -cube,  $\mathcal{R}$  is a binary relation ( $\Pi x : A. B$  has the same sorts as  $B$ ).

(ii) An extension of  $\lambda\omega$  that supports “polymorphic identity function on types”, where

- (a)  $\mathcal{S} = \{\star, \square, \square'\}$

(Axiom)	$\frac{}{\vdash s : t}$	$(s, t) \in \mathcal{A}$
(Var)	$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$	$x \notin \text{dom}(\Gamma)$
(Weak)	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B}$	$x \notin \text{dom}(\Gamma)$
(App)	$\frac{\Gamma \vdash f : (\Pi x : A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash fa : B[x := a]}$	
(Lam)	$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A. B) : t}{\Gamma \vdash (\lambda x : A. b) : (\Pi x : A. B)}$	
(Pi)	$\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : t}{\Gamma \vdash (\Pi x : A. B) : u}$	$(s, t, u) \in \mathcal{R}$
(Conv)	$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : s \quad A =_{\beta} B}{\Gamma \vdash a : B}$	

**Table 1.** Typing rules for pure type system

(b)  $\mathcal{A} = \{(\star, \square), (\square, \square')\}$

(c)  $\mathcal{R} = \{(\star, \star), (\square, \star), (\square, \square), (\square', \square')\}$

in which we can have  $\vdash (\lambda \kappa : \square. \lambda \alpha : \kappa. \alpha) : \Pi \kappa : \square. \kappa \rightarrow \kappa$ .

### 3. Typing Derivations

Examples of typing derivations go here.

### References

- [1] Simon Peyton Jones and Erik Meijer. Henk: a typed intermediate language. *TIC*, 97, 1997.
- [2] J-W Roorda and JT Jeuring. Pure type systems for functional programming. 2007.
- [3] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149. Elsevier, 2006.