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* Γ are defined by

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

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) Π and λ are used to bind variables. Let $\mathrm{FV}(A)$ 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 $A \to B$ as $(\Pi x : A.B)$ where $x \notin \mathrm{FV}(B)$.

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

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

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 λ-cube consists of eight PTSs, where

  (a) $\mathcal{S} = \{\star, \Box\}$

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

  (c) $\{(\star, \star)\} \subseteq \mathcal{R} \subseteq \{(\star, \star), (\star, \Box), (\Box, \star), (\Box, \Box)\}$

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 λ-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) $\mathcal{S} = \{\star, \Box, \Box'\}$

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

Table 1. Typing rules for pure type system

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

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

in which we can have $\vdash (\lambda\kappa : \Box.\lambda\alpha : \kappa.\alpha) : (\Pi\kappa : \Box.\kappa \to \kappa)$ (justified in Section 3).

3. Typing Derivations

The typing derivation of $\vdash (\lambda\kappa : \Box.\lambda\alpha : \kappa.\alpha) : (\Pi\kappa : \Box.\Pi\alpha : \kappa.\kappa)$ is as follows:

$$\dfrac{\dfrac{\dfrac{\mathrm{B}}{\kappa : \Box, \alpha : \kappa \vdash \alpha : \kappa}\,\mathit{Var} \qquad \mathrm{A}}{\kappa : \Box \vdash (\lambda\alpha : \kappa.\alpha) : (\Pi\alpha : \kappa.\kappa)}\,\mathit{Lam} \qquad \dfrac{\dfrac{}{\vdash \Box : \Box'}\,\mathit{Ax} \qquad \mathrm{A}}{\vdash (\Pi\kappa : \Box.\Pi\alpha : \kappa.\kappa) : \Box}\,\mathit{Pi}}{\vdash (\lambda\kappa : \Box.\lambda\alpha : \kappa.\alpha) : (\Pi\kappa : \Box.\Pi\alpha : \kappa.\kappa)}\,\mathit{Lam}$$

where

$$A = \quad \dfrac{\mathrm{B} \qquad \dfrac{\mathrm{B} \qquad \mathrm{B}}{\kappa : \Box, \alpha : \kappa \vdash \kappa : \Box}\,\mathit{Weak}}{\kappa : \Box \vdash (\Pi\alpha : \kappa.\kappa) : \Box}\,\mathit{Pi}$$

$$B = \quad \dfrac{\dfrac{}{\vdash \Box : \Box'}\,\mathit{Ax}}{\kappa : \Box \vdash \kappa : \Box}\,\mathit{Var}$$

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.