

## 2006 Paper 9 Question 11

### Types

Given any polymorphic lambda calculus (PLC) type  $\tau$  and any function  $\rho$  mapping type variables  $\alpha$  to values  $n \in \{-1, 0, 1\}$ , a value  $\llbracket \tau \rrbracket \rho$  in  $\{-1, 0, 1\}$  is defined by recursion on the structure of  $\tau$  as follows.

$$\llbracket \alpha \rrbracket \rho = \rho(\alpha)$$

$$\llbracket \tau_1 \rightarrow \tau_2 \rrbracket \rho = \begin{cases} 1 & \text{if } \llbracket \tau_1 \rrbracket \rho \leq \llbracket \tau_2 \rrbracket \rho \\ \llbracket \tau_2 \rrbracket \rho & \text{otherwise} \end{cases}$$

$\llbracket \forall \alpha (\tau) \rrbracket \rho =$  the minimum of the values  $\llbracket \tau \rrbracket (\rho[\alpha \mapsto n])$  for  $n = -1, 0, 1$  (where  $\rho[\alpha \mapsto n]$  is the function mapping  $\alpha$  to  $n$  and every other  $\alpha'$  to  $\rho(\alpha')$ ).

If  $\Gamma$  is a non-empty PLC typing environment, let  $\llbracket \Gamma \rrbracket \rho$  denote the minimum value of  $\llbracket \tau \rrbracket \rho$  as  $\tau$  ranges over the types in  $\Gamma$ ; in the case that  $\Gamma$  is empty, we define  $\llbracket \Gamma \rrbracket \rho$  to be 1.

- (a) Prove that if  $\Gamma \vdash M : \tau$  is a valid PLC typing judgement, then for any  $\rho$ ,  $\llbracket \Gamma \rrbracket \rho \leq \llbracket \tau \rrbracket \rho$ .

You may assume without proof that if  $\alpha$  is not free in  $\tau$  then

$$\llbracket \tau \rrbracket (\rho[\alpha \mapsto n]) = \llbracket \tau \rrbracket \rho$$

and also that type substitutions  $\tau[\tau'/\alpha]$  satisfy

$$\llbracket \tau[\tau'/\alpha] \rrbracket \rho = \llbracket \tau \rrbracket (\rho[\alpha \mapsto \llbracket \tau' \rrbracket \rho])$$

[Hint: show that the property  $\Phi(\Gamma, M, \tau) =$  “for all  $\rho$ ,  $\llbracket \Gamma \rrbracket \rho \leq \llbracket \tau \rrbracket \rho$ ” is closed under the rules of the typing system.]

[16 marks]

- (b) Deduce that there is no closed PLC expression of type

$$\forall \alpha, \beta ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$$

[4 marks]