

## 2010 Paper 6 Question 10A

### Semantics of Programming Languages

The version of this question shown in the full paper is as it was originally printed. A corrected version of the question is given here.

Below is the syntax and operational semantics for a pure functional language.

*Types:*  $T ::= \text{bool} \mid T \rightarrow T$   
*Variables:*  $\{x, y, z, \dots\}$   
*Expressions:*  $e ::= \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \text{fn}(x : T) \Rightarrow e \mid e e'$ .

In the expression  $\text{fn}(x : T) \Rightarrow e$ , the variable  $x$  is binding in  $e$ .

$$\begin{aligned} \text{(if1)} \quad & (\text{if true then } e_1 \text{ else } e_2) \longrightarrow e_1 \\ \text{(if2)} \quad & (\text{if false then } e_1 \text{ else } e_2) \longrightarrow e_2 \\ \text{(if3)} \quad & \frac{e \longrightarrow e'}{(\text{if } e \text{ then } e_1 \text{ else } e_2) \longrightarrow (\text{if } e' \text{ then } e_1 \text{ else } e_2)} \\ \text{(app)} \quad & \frac{e_1 \longrightarrow e'_1}{e_1 e_2 \longrightarrow e'_1 e_2} \\ \text{(fn)} \quad & (\text{fn}(x : T) \Rightarrow e) e' \longrightarrow \{e'/x\}e \end{aligned}$$

(There is no need for a store because there are no store access operations.)

- (a) Is this a call-by-value or a call-by-name language? Revise the operational semantics to demonstrate the other calling convention. [4 marks]
- (b) A type environment is a finite partial function  $\Gamma$  from variables to types. Define a typing relation  $\Gamma \vdash e : T$  by giving a set of rules. [6 marks]
- (c) Are the following expressions typable?

$$\begin{aligned} e_1 &= \text{fn}(f : (\text{bool} \rightarrow \text{bool}) \rightarrow \text{bool}) \Rightarrow (\text{fn}(f : \text{bool} \rightarrow \text{bool}) \Rightarrow f f) \\ e_2 &= \text{fn}(f : \text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})) \Rightarrow (\text{fn}(x : \text{bool}) \Rightarrow (f x) x) \end{aligned}$$

[2 marks]

- (d) State formally the following two theorems of the one-step reduction semantics at the top of the page and the type system that you defined in part (b): Progress and Type Preservation. Take care to explain what a value is. (No proofs are required for this part.) [3 marks]
- (e) State and prove the Type Safety theorem. You may use the results stated in part (d). [5 marks]