8 Semantics of Programming Languages (PMS)

Consider the following syntax up to alpha equivalence, where $n$ ranges over natural numbers, $x$ over a set of variables, and (as usual) $x$ is binding in $e$ in $\text{fn } x \Rightarrow e$.

**expressions**, $e ::= n \mid x \mid \text{fn } x \Rightarrow e \mid e \cdot e'$

**values**, $v ::= n \mid x \mid \text{fn } x \Rightarrow e$

(a) Define free variables $\text{fv}(e)$ and capture-avoiding substitution $\{e/z\} e'$. [3 marks]

(b) Define a left-to-right call-by-value reduction relation $e \rightarrow e'$. [3 marks]

Implementing a language using substitution is inefficient, as each substitution has to traverse a potentially large subterm. Consider the following proposal for an abstract machine for this language using environments $E$, lists of variable/value pairs.

\[
\langle E, e \rangle \rightarrow \langle E', e' \rangle
\]

(c) Give the sequence of abstract-machine reduction steps, including the configurations and the names of the rules used, for the initial configuration below. You need not give full derivation trees.

\[
\langle [], ((\text{fn } x \Rightarrow (\text{fn } y \Rightarrow xy)) (\text{fn } z \Rightarrow z)) 3 \rangle
\]

[5 marks]

(d) Explain, with a concrete example and its reduction sequence, what could go wrong if the premise of $\text{fn}$ had been omitted. [5 marks]

(e) Write $\{E\} e$ for the iterated substitution defined by

\[
\{[]\} e = e \quad \{(x, v):: E\} e = \{E\}(\{v/x\} e)
\]

Prove that $\{E\}(e_1 e_2) = (\{E\}e_1 \{E\} e_2)$. [4 marks]