10 Semantics of Programming Languages (PMS)

Let x range over a set X of identifiers, n range over the natural numbers $\mathbb{N}$, and s range over stores: total functions from X to $\mathbb{N}$.

Consider a language with the following abstract syntax.

$e ::= \quad n \mid x := e \mid !x \mid e_1; e_2$

(*a*) Define a conventional deterministic small-step operational semantics $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ for the language. Comment briefly on the choices you make. [5 marks]

(*b*) If your language is deterministic and terminating, the operational semantics implicitly defines a more abstract semantics: we can regard each expression as a function over stores $[\![e]\!]$ that takes store s to the unique number n and store s' such that

$$\langle e, s\rangle \longrightarrow^* \langle n, s'\rangle \wedge \nexists e'', s''. \ \langle n, s'\rangle \longrightarrow \langle e'', s''\rangle$$

This language is quite limited in expressiveness. Describe, as clearly and precisely as you can, the set of functions from stores to (number, store) pairs that are expressible as $[\![e]\!]$ for some e. [5 marks]

(*c*) The primitive contexts C for this language are expressions with a single hole:

$C ::= \quad x := \_ \mid e_1; \_ \mid \_; e_2$

Write $C[e]$ for the expression resulting from replacing the hole in C by e.

Say a binary relation $\sim$ over expressions is a congruence if $e \sim e'$ implies $\forall C.\ C[e] \sim C[e']$.

Say a binary relation $\sim$ over expressions respects final values if $e \sim e'$ implies $\forall s_0, n, n', s, s'.\ (\langle e, s_0\rangle \longrightarrow \langle n, s\rangle \wedge \langle e', s_0\rangle \longrightarrow \langle n', s'\rangle) \Rightarrow n = n'$.

Use your characterisation of part (*b*) to define an equivalence relation over expressions that is a congruence and respects final values. Explain briefly why it has those properties. [4 marks]

(*d*) Define a terminating algorithm that, for any expressions e and e', computes whether $e \sim e'$ or not. Explain informally why it is correct. *Hint:* you may want to adapt your semantics from part (*a*) to compute symbolically.

[6 marks]