1999 Paper 5 Question 9

Semantics of Programming Languages

The commands C of a language for manipulating integer storage locations ℓ are given by

$$\begin{array}{rl} C ::= & \textbf{skip} \mid \ell := E \mid C \,;\, C \mid \textbf{if}\ B\ \textbf{then}\ C\ \textbf{else}\ C \\ & \mid \textbf{while}\ B\ \textbf{do}\ C \mid \textbf{begin loc}\,\ell := E \,;\, C\ \textbf{end} \end{array}$$

where B and E range over boolean and integer expressions respectively, whose precise syntax you do not need to specify, but which include various integer and boolean operations and an operation for reading the contents of a location. The last form of command is for block-structured local state.

You may assume that evaluation of integer expressions E to integer values n has no side-effects on states s and that a suitable evaluation relation of the form $E, s \Downarrow n$ has already been defined; similarly an evaluation relation $B, s \Downarrow b$ for boolean expressions has already been defined. Using these relations, give an operational semantics for commands in the form of an inductively defined relation $C, s \Downarrow s'$ for evaluating a command C in a state s, resulting in a final state s'. Make sure that your definition treats properly the local scope of ℓ in a block $\textbf{begin loc}\,\ell := E \,;\, C\ \textbf{end}$. Illustrate this by showing that evaluating $\ell := 0 \,;\, (\textbf{begin loc}\,\ell := 1 \,;\, \textbf{skip end})$ in any state results in a state in which the value stored in ℓ is 0, not 1. [12 marks]

Use the operational semantics to give a definition of *semantic equivalence* for these commands. Prove that if $\ell \neq \ell'$ then $\textbf{begin loc}\,\ell := E \,;\, \ell' := !\ell\ \textbf{end}$ and $\ell' := E$ are semantically equivalent for any integer expression E. What happens if $\ell = \ell'$? If C does not involve any occurrences of ℓ, is $\textbf{begin loc}\,\ell := E \,;\, C\ \textbf{end}$ always semantically equivalent to C? [8 marks]