

1996 Paper 5 Question 12

Semantics

The abstract syntax of commands in a simple parallel programming language P is given by

$$C ::= \text{skip} \mid X := ie \mid C_1 ; C_2 \mid \text{if } be \text{ then } C_1 \text{ else } C_2 \mid \text{while } be \text{ do } C \mid C_1 \parallel C_2$$

where ie , be and X range over the syntactic categories of integer expressions, boolean expressions and program variables, respectively. The intended behaviour of $C_1 \parallel C_2$ is that C_1 and C_2 are executed in parallel until they have both terminated. Hence atomic execution steps from C_1 and C_2 may be arbitrarily interleaved. The other command forms behave as usual.

- (a) Give a small-step transition semantics for P which derives statements of the form $\langle C, S \rangle \rightarrow \langle C', S' \rangle$, where S and S' are states. You may assume that rules for the evaluation of expressions have already been given.

Comment briefly on your choice of what constitutes an atomic execution step. [9 marks]

- (b) The binary relation \sim on commands is defined by

$$C_1 \sim C_2 \equiv \forall S, S'. \langle C_1, S \rangle \rightarrow^* \langle \text{skip}, S' \rangle \iff \langle C_2, S \rangle \rightarrow^* \langle \text{skip}, S' \rangle.$$

Show that \sim is *not* a congruence. [5 marks]

- (c) Assuming that $S(X) = S(Y) = 0$, describe the set of possible execution traces which are derivable in your semantics starting from the configuration $\langle C, S \rangle$, where C is

$$(X := 1) \parallel (\text{while } X = 0 \text{ do } Y := Y + 1).$$

Why might one argue that this does not accurately reflect the behaviour of a reasonable implementation of the language? [6 marks]