Due: Friday 30 January 2004 3pm

Exercise 1

Draw the complete inference tree for the following statement

\[ \langle x := 1; \text{while } x < y \text{ do } x := 2 \times x, \sigma \rangle \rightarrow \sigma' \]

where \( \sigma(x) = 0 \) and \( \sigma(y) = 3 \). Explain what \( \sigma' \) is.

Exercise 2

Let \( c_0 \sim c_1 \) be defined by: for all \( \sigma, \sigma' \in \Sigma, \langle c_0, \sigma \rangle \rightarrow \sigma' \) iff \( \langle c_1, \sigma \rangle \rightarrow \sigma' \). Prove:

a) if \( b \) then \( c \) else \( c \sim c \).

b) while \( b \) do \( c \sim \) if \( b \) then \( (c; \text{while } b \text{ do } c) \) else skip.

You may use, without a proof, the totality of the transition system for Boolean expressions. That is, you can assume that for any \( b \in \text{Bexp} \) and any state \( \sigma \) one can infer either \( \langle b, \sigma \rangle \rightarrow \text{true} \) or \( \langle b, \sigma \rangle \rightarrow \text{false} \).

Exercise 3

Using the operational semantics of Imp prove that for any program \( c \in \text{Com} \) which does not contain any assignment statements if \( \langle c, \sigma \rangle \rightarrow \sigma' \) then \( \sigma' = \sigma \).

Imagine introducing the break command into Imp, whose intended behaviour is to immediately abort execution of the smallest enclosing while-loop (if any) and to return control to the following commands (if any).

Thus for example, using this new command, the command while \( b \) do \( c \) is equivalent to

while true do if \( b \) then \( c \) else break

The syntax of the extended language is

\[ c ::= \text{skip} \mid \text{break} \mid x := a \mid c_0;c_1 \mid \text{if } b \text{ then } c_0 \text{ else } c_1 \mid \text{while } b \text{ do } c. \]

Exercise 4

Give a complete operational semantic definition for this extended language.

Note: You only need to give rules for executing commands—infERENCE rules for arithmetic and Boolean expressions are the same as in the original Imp language.