Series 1

Exercise 1

Consider the following statement \((z := x; x := y); y := z,\) and the environment \(\sigma_0\) which maps every variables but \(x\) and \(y\) to 0, maps \(x\) to 5, and \(y\) to 7. Give a derivation tree of this statement.

Exercise 2

We consider the arithmetical expressions defined in the course lecture.
Let \(a, a' \in A_{\text{exp}}\), and \(\sigma, \sigma'\) two states. Let \(X\) be the set of variables appearing in \(a\).
1. Prove that if \(\forall x \in X \cdot \sigma(x) = \sigma'(x)\), then \(A[a]\sigma = A[a]\sigma'\).
2. Prove that \(A[a[a'/x]]\sigma = A[a]\sigma[x \mapsto A[a']\sigma]\).

Exercise 3

We consider the following statements:
\begin{itemize}
\item while \(\neg(x = 1)\) do \((y := y \ast x; x := x - 1)\) od
\item while \(1 \leq x\) do \((y := y \ast x; x := x - 1)\) od
\item while true do skip od
\end{itemize}
where \(x\) designates a variable of type \(\mathbb{Z}\).
For each of the preceding statement, determine whether :
1. its execution loops in every state
2. its execution stops in every state
3. there are states from which the execution terminates, and some from which it does not.
Prove your answers.

Exercise 4

We wish to add the following statement to the While language:

\[
\text{repeat } S \text{ until } b
\]

1. Provide the semantics rules in order to define \(\text{repeat } S \text{ until } b\) without using the \(\text{while } b \text{ do } \cdots \text{ od}\) construction.
2. Prove that
   \[(a) \text{ repeat } S \text{ until } b\]
(b) \( S; \text{if } b \text{ then skip else (repeat } S \text{ until } b) \).

are semantically equivalent.

3. We want to prove that the statement repeat \( S \) until \( b \) does not add expressive power. To do so, give a function which transforms every program with the statement repeat \( S \) until \( b \) into a program in the While language. Is the given transformation computable? Compare the size of a program and its image resulting of the transformation.

**Exercise 5**

Prove that, for all statements \( S_1, S_2, S_3 \), the following statements are semantically equivalent:

1. \( S_1; (S_2; S_3) \)
2. \( (S_1; S_2); S_3 \)

Prove that, in general, \( S_1; S_2 \) is not semantically equivalent to \( S_2; S_1 \).

**Exercise 6**

Prove that the natural operational semantics of the While language is deterministic.

**Exercise 7**

Define the semantics of the set of boolean expressions \( \text{BExp} \).

**Exercise 8**

We build a set \( B \) of boolean expressions using the following elements:

- constants \( true \) and \( false \),
- a set of boolean variables denoted \( \text{Bool} \)
- \( \neg \) rule: if \( b \in B \) then \( \neg b \in B \)
- \( \land \) rule: if \( b_1, b_2 \in B \) then \( b_1 \land b_2 \in B \)

Write the formal sentence corresponding to the following english sentence and prove it: two states that coincide on every boolean variable yield equal values for any expression in \( B \).

Optional question: how can we adapt this statement for \( \text{Bexp} \)? A set \( B \) of \( \text{BExp} \), set of boolean expressions, can be constructed inductively from atoms \( true \) and \( false \) and using the following rules:

- \( \neg \) rule: if \( b \in B \) then \( \neg b \in B \)
- \( \land \) rule: if \( b_1, b_2 \in B \) then \( b_1 \land b_2 \in B \)

Write the formal sentence corresponding to the following english sentence and prove it: two states that coincide on every boolean variable yield equal values for any expression in \( B \).

Optional question: is it true for any expression in \( \text{Bexp} \)?

**Exercise 9**
We consider the language defined by the following BNF:

\[
S ::= x := a \mid \text{skip} \mid S_1 ; S_2 \mid \\
\text{if } b \text{ then } S_1 \text{ else } S_2
\]

What can we say about termination of programs written in this language (according to the natural semantics)? Prove your statement.