

Aufgabe 1 : (4+1 Punkte)

Unter den *freien Variablen* eines Booleschen Ausdrucks b verstehen wir die Menge der in ihm vorkommenden Variablen. Diese Menge lässt sich induktiv wie folgt definieren:

$$\begin{aligned}FV(\text{true}) &= \emptyset \\FV(\text{false}) &= \emptyset \\FV(a_1 = a_2) &= FV(a_1) \cup FV(a_2) \\FV(a_1 \leq a_2) &= FV(a_1) \cup FV(a_2) \\FV(\neg b_1) &= FV(b_1) \\FV(b_1 \wedge b_2) &= FV(b_1) \cup FV(b_2) \\FV(b_1 \vee b_2) &= FV(b_1) \cup FV(b_2)\end{aligned}$$

1. Beweisen Sie induktiv: Sind σ und σ' zwei Zustände mit $\sigma(x) = \sigma'(x)$ für alle $x \in FV(b)$, dann gilt:

$$\llbracket b \rrbracket_B(\sigma) = \llbracket b \rrbracket_B(\sigma')$$

2. Was bedeutet die vorstehende Aussage anschaulich?

Aufgabe 2 : (5+5 Punkte)

Beweisen Sie mithilfe der

1. SO- und
2. N-Semantik

dass die sequentielle Komposition assoziativ ist, d.h. die Anweisungsfolgen

$$\pi_1; (\pi_2; \pi_3) \quad \text{und} \quad (\pi_1; \pi_2); \pi_3$$

sind semantisch äquivalent.

Aufgabe 3 : (5+5 Punkte)

Sei $\sigma \in \Sigma$ ein Zustand mit $\sigma(x) = 13$ und $\sigma(y) = 5$. Zeigen Sie mithilfe der

1. strukturell operationellen
2. natürlichen

Semantik von WHILE, dass das Programm

$$z := 0; \text{ while } y \leq x \text{ do } z := z + 1; x := x - y \text{ od}$$

angesetzt auf σ regulär im Zustand $\sigma[2/z][5/y][3/x]$ terminiert.

Aufgabe 4 : (5 Punkte)

Seien $\pi_1, \pi_2 \in \mathbf{Prg}$ und $\sigma, \sigma' \in \Sigma$.

Untersuchen Sie die Gültigkeit der folgenden Implikation (Beweis oder Gegenbeispiel):

$$\langle \pi_1; \pi_2, \sigma \rangle \Rightarrow^* \langle \pi_2, \sigma' \rangle \succ \exists k \in \mathbb{N}_0. \langle \pi_1, \sigma \rangle \Rightarrow^k \sigma'$$

Aufgabe 5 : (5+5 Punkte)

Wir erweitern die Programmiersprache WHILE um das Konstrukt

`repeat π until b end`

Geben Sie eine

1. SOS-Regel [rep_sos]
2. NS-Regel [rep_ns]

an, die diesem Konstrukt die “gewohnte” Semantik gibt, ohne bei der Angabe dieser Regeln die Existenz des while-Konstrukts in WHILE auszunutzen.

Aufgabe 6 : (5+5 Punkte)

Wir erweitern die Programmiersprache WHILE um das Konstrukt

`for $x := a_1$ to a_2 do π od`

Definieren Sie die SO- und N-Semantik für dieses Konstrukt so, dass es der “gewohnten” Bedeutung der for-Schleife entspricht.