Theoretische Informatik 2 – Übungsblatt 6 mit Musterlösungen

(LV-Nr.: 185.183, 185.231, WS 2004/2005)

Thema: Formale Aspekte von Progr.sprachen 2 – Operationale Semantik

Übung 6.1 *Geben Sie ein **WHILE**-Programm P an mit der Eigenschaft, dass P zu $P' = P; P$ nicht semantisch äquivalent ist (bzgl. natürlicher Semantik). Beweisen Sie die Nichtäquivalenz von P und P'.*

Lösung: Wir wählen $P = x := x + 1$. Dann erhalten wir $P' = x := x + 1; x := x + 1$. Zwei Programme P_1 und P_2 sind *semantisch äquivalent*, falls für alle Zustände s, s' gilt:

$$\langle P_1, s\rangle \to s' \quad \text{gdw.} \quad \langle P_2, s\rangle \to s' \,.$$

In unserem Fall erhalten wir

$$\langle x := x + 1, s\rangle \to s[x \mapsto sx + 1] \; [\mathsf{ass}_{\mathsf{ns}}]$$

sowie

$$\frac{\langle x := x + 1, s\rangle \to s[x \mapsto sx + 1] \quad \langle x := x + 1, s[x \mapsto sx + 1]\rangle \to s[x \mapsto sx + 2]}{\langle x := x + 1; x := x + 1, s\rangle \to s[x \mapsto sx + 2]} \; [\mathsf{comp}_{\mathsf{ns}}]$$

Folglich gilt: $\langle P, s\rangle \to s'$ für $s' = s[x \mapsto sx + 1]$ und $\langle x := x + 1; x := x + 1, s\rangle \to s''$ für $s'' = s[x \mapsto sx + 2]$. Wegen $s'x = sx + 1 \neq sx + 2 = s''x$ folgt $s' \neq s''$, also sind P und P' nicht semantisch äquivalent (in NS).

Übung 6.2 *Gegeben seien die beiden **WHILE**-Programme*

$$P_1 \;=\; (x := 2; \mathsf{while}\ x \leq 2\ \mathsf{do}\ x := x + 1)$$

und

$$P_2 \;=\; x := 3 \,.$$

Zeigen Sie, dass P_1 und P_2 (in struktureller operationaler Semantik) semantisch äquivalent sind.

Lösung: Für einen beliebigen Anfangszustand s erhalten wir $\langle P_2 \;=\; x := 3, s\rangle \Rightarrow s[x \mapsto 3]$ mittels $[\mathsf{ass}_{\mathsf{sos}}]$ sowie

$$\begin{array}{lll}
\langle P_2, s\rangle & \Rightarrow & \langle \mathsf{while}\ x \leq 2\ \mathsf{do}\ x := x + 1, s[x \mapsto 2]\rangle \\
& \Rightarrow & \langle \mathsf{if}\ x \leq 2\ \mathsf{then}\ (x := x + 1; \mathsf{while}\ x \leq 2\ \mathsf{do}\ x := x + 1)\ \mathsf{else}\ \mathsf{skip}, s[x \mapsto 2]\rangle \\
& \Rightarrow & \langle x := x + 1; \mathsf{while}\ x \leq 2\ \mathsf{do}\ x := x + 1, s[x \mapsto 2]\rangle \\
& \Rightarrow & \langle \mathsf{while}\ x \leq 2\ \mathsf{do}\ x := x + 1, s[x \mapsto 3]\rangle \\
& \Rightarrow^* & s[x \mapsto 3]
\end{array}$$

(mit entsprechenden Rechtfertigungen der Einzelschritte). In beiden Fällen gibt es keine anderen anderen *hängenden* oder *End*konfigurationen und auch keine nichtterminierenden Ableitungen, jeweils ausgehend von $\langle P_1, s\rangle$ bzw. $\langle P_2, s\rangle$. Folglich sind P_1und P_2 (in struktureller operationaler Semantik) semantisch äquivalent.

Übung 6.3 *Gegeben sei das* ***WHILE***$^{\mathsf{or}}$*-Programm*

$$P \;=\; \mathsf{while}\ x \leq y\ \mathsf{do}\ (x := 20\ \mathsf{or}\ y := y - 1)$$

(in der um das nichtdeterministische Konstrukt or *erweiterten Sprache* ***WHILE****) und ein Anfangszustand s mit $sx = -5$, $sy = 10$. Berechnen Sie gemäß der erweiterten Übergansgrelation für* ***WHILE***$^{\mathsf{or}}$ *die Menge Z alle Folgezustände, d.h.*

$$Z = \{s' \mid \langle P, s\rangle \to s'\}\ .$$

Lösung: Wir verwenden folgende Abkürzungen für Programm-Teile:

$$P \;=\; \mathsf{while}\ \underbrace{x \leq y}_{b}\ \mathsf{do}\ \underbrace{(\overbrace{x := 20}^{P_1}\ \mathsf{or}\ \overbrace{y := y - 1}^{P_2})}_{S}$$

Gemäß I_{ns}, erweitert um die or-Regeln, erhalten wir:

$$\dfrac{\dfrac{[\mathsf{ass}_{\mathsf{ns}}]}{\langle P_1, s\rangle \to s'}}{\langle S, s\rangle \to s'}\ [\mathsf{or}^1_{\mathsf{ns}}] \qquad \dfrac{[\mathsf{while}^{\mathbf{f}}_{\mathsf{ns}}]}{\langle P, s'\rangle \to s'}}{\langle P, s\rangle \to s'}\ [\mathsf{while}^{\mathbf{t}}_{\mathsf{ns}}]$$

wobei $s' = s[x \mapsto 20]$, da $\mathcal{B}[\![b]\!]s' = 20 \leq -5 = \mathbf{f}$. Eine andere Möglichkeit ergibt sich durch anfängliche Auswahl der rechten or-Alternative:

$$\dfrac{\dfrac{\dfrac{[\mathsf{ass}_{\mathsf{ns}}]}{\langle P_2, s\rangle \to s[y \mapsto 9]}}{\langle S, s\rangle \to s[y \mapsto 9]}\ [\mathsf{or}^2_{\mathsf{ns}}] \qquad \dfrac{\dfrac{\dfrac{[\mathsf{ass}_{\mathsf{ns}}]}{\langle P_1, s[y \mapsto 9]\rangle \to s''}}{\langle S, s[y \mapsto 9]\rangle \to s''}\ [\mathsf{or}^1_{\mathsf{ns}}] \qquad \dfrac{[\mathsf{while}^{\mathbf{f}}_{\mathsf{ns}}]}{\langle P, s''\rangle \to s''}}{\langle P, s[y \mapsto 9]\rangle \to s''}\ [\mathsf{while}^{\mathbf{t}}_{\mathsf{ns}}]}{\langle P, s\rangle \to s''}\ [\mathsf{while}^{\mathbf{t}}_{\mathsf{ns}}]$$

wobei $s'' = s[y \mapsto 9][x \mapsto 20] = s[x \mapsto 20][y \mapsto 9]$, da $\mathcal{B}[\![b]\!]s[y \mapsto 9][x \mapsto 20] = 20 \leq 9 = \mathbf{f}$. Offensichtlich wird die while-Schleife beliebig durchlaufen, bis der Wert von x größer als der Wert von y ist. Letzeres ist immer sofort dann der Fall, wenn der linke "or-Zweig" gewählt wird, oder spätestens dann, wenn y, durch wiederholte Auswahl des rechten "or-Zweiges", zu -6 wird. Folglich ist die Menge Z aller Endergebnisse wir folgt bestimmt:

$$Z = \{s' \mid s' = s[x \mapsto 20][y \mapsto k], -5 \leq k \leq 10\} \cup \{s[y \mapsto -6]\}\ .$$

Übung 6.4 *Gegeben sei das* ***WHILE***$^{\mathsf{par}}$*-Programm*

$$P\colon (y := 3; y := y - 1)\ \mathsf{par}\ (y := 5; y := y * 4)$$

und ein beliebiger Anfangszustand s. Betrachten Sie alle Möglichkeiten für

(a) $\langle P, s\rangle \to s'$ *bzw.*

(b) $\langle P, s\rangle \Rightarrow^* s'$

in dem jeweils um die Parallelismus-Regeln erweitern Inferenzsystem (für NS/SOS) und geben Sie in beiden Fällen alle möglichen End-Werte $s'y$ an.

Lösung: Notation:

$$P\colon \overbrace{(\underbrace{y := 3}_{a}; \underbrace{y := y - 1}_{b})}^{e} \mathsf{par}\ \overbrace{(\underbrace{y := 5}_{c}; \underbrace{x := y := y * 4}_{d})}^{f}.$$

Die möglichen Werte von $s'y$ (in NS und SOS) sind:

NS:

Reihenfolge	$s'y$
e f	20
f e	2

SOS:

Reihenfolge	$s'y$
a b c d	20
a c b d	16
a c d b	19
c a b d	8
c a d b	11
c d a b	2

Beispielsweise entspricht die erste Zeile in der NS-Tabelle der Ableitung

$$\dfrac{\dfrac{[\mathsf{ass}_{\mathsf{ns}}] \quad [\mathsf{ass}_{\mathsf{ns}}]}{\langle a, s\rangle \to s_2 \quad \langle b, s_2\rangle \to s_1}\ [\mathsf{comp}_{\mathsf{ns}}] \quad \dfrac{[\mathsf{ass}_{\mathsf{ns}}] \quad [\mathsf{ass}_{\mathsf{ns}}]}{\langle c, s_1\rangle \to s_3 \quad \langle d, s_3\rangle \to s'}\ [\mathsf{comp}_{\mathsf{ns}}]}{\langle P, s\rangle \to s'}\ [\mathsf{par}^{\mathsf{lr}}_{\mathsf{ns}}]$$

mit $s' = s[y \mapsto 20]$, $s_1 = s[y \mapsto 2]$, $s_2 = s[y \mapsto 3]$, und $s_3 = s[y \mapsto 5]$.

Die vierte Zeile der SOS-Tabelle (cabd) etwa entspricht der Ableitung (hier: ohne jeweilige Rechtfertigungen)

$$\begin{array}{lll} \langle P, s\rangle & \overset{[\mathsf{par}^{\mathsf{rc}}_{\mathsf{sos}}]}{\Rightarrow} & \langle c\ \mathsf{par}\ d, s_1\rangle \\ & \overset{[\mathsf{par}^{\mathsf{lc}}_{\mathsf{sos}}]}{\Rightarrow} & \langle b\ \mathsf{par}\ d, s_2\rangle \\ & \overset{[\mathsf{par}^{\mathsf{la}}_{\mathsf{sos}}]}{\Rightarrow} & \langle d, s_3\rangle \\ & \overset{[\mathsf{ass}_{\mathsf{sos}}]}{\Rightarrow} & s' \end{array}$$

mit $s' = s[y \mapsto 8]$, $s_1 = s[y \mapsto 5]$, $s_2 = s[y \mapsto 3]$, $s_3 = s[y \mapsto 2]$, $s' = s[y \mapsto 8]$.

Übung 6.5 *Sei* ***Prog′*** *die Menge aller* ***WHILE****-Programme, die keine* while*-Schleife enthalten. Zeigen Sie: Alle Programme aus* ***Prog′*** *terminieren (gemäß natürlicher Semantik). Hinweis: Verwenden Sie strukturelle Induktion über den Aufbau von Programmen aus* ***Prog′****.*

Lösung: Wir beweisen

$$(*)\ \ \forall P \in \mathbf{Prog}'\ \forall s\ \exists s' : \langle P, s\rangle \to s'$$

durch struktuelle Induktion über den Aufbau von Programm aus $\mathbf{Prog}'$.
Induktionsanfang: Für Zuweisung und Leeranweisung (skip) ist die Behauptung offensichtlich erfüllt.
Induktionsschluss: Wir haben 3 Fälle zu betrachten:

- Komposition: $P = P_1; P_2$

  $$\dfrac{\langle P_1, s\rangle \to s' \quad \langle P_2, s'\rangle \to s''}{\langle P_1; P_2, s\rangle \to s''}\ [\mathsf{comp}_{\mathsf{ns}}]$$

  Nach IH existieren s' und s'' mit $\langle P_1, s\rangle \to s'$ und $\langle P_2, s'\rangle \to s''$, folglich gilt auch $\langle P_1; P_2, s\rangle \to s''$.

- Die beiden anderen Fälle für bedingte Anweisungen ($[\mathsf{if}^{\mathbf{t}}_{\mathsf{ns}}]$ und $[\mathsf{if}^{\mathbf{f}}_{\mathsf{ns}}]$) gehen analog.

Damit ist der Induktionsbeweis von (*) fertig, und die Behauptung bewiesen.
(Man beachte, dass die beiden Inferenzregeln für while-Schleifen hier nicht relevant sind, da die Programme aus $\mathbf{Prog}'$ keine while-Schleifen enthalten!)