3 Operationale Semantik für While

Eine operationale Semantik für While beschreibt nicht nur das Ergebnis einer Programmausführung, sondern auch, wie man zu diesem Ergebnis gelangen kann. Dafür gibt es zwei Modellierungsansätze:

Big-step Semantik (Auswertungssemantik, natural semantics): Hier wird beschrieben, wie man aus dem Verhalten der Teile eines Programmkonstrukts dessen Gesamtverhalten konstruiert.

Small-step Semantik (Transitionssemantik, structural operational semantics): Hier liegt der Fokus auf einzelnen, kleinen Berechnungsschritten, die nach vielen Schritten zum Ende der Programmausführung gelangen.

3.1 Big-Step-Semantik für While

Eine Big-Step Semantik ist eine Auswertungsrelation $\langle c,\ \sigma\rangle \Downarrow \sigma'$, die für ein Programm c und einen Anfangszustand σ bestimmt, ob σ' ein möglicher Endzustand einer Ausführung von c in σ ist.

Definition 4 (Big-Step-Semantik).
Die Auswertungsrelation $\langle c,\ \sigma\rangle \Downarrow \sigma'$ wird durch folgende Regeln induktiv definiert:

$$\textsc{Skip}_{\text{BS}}\colon \langle \texttt{skip},\ \sigma\rangle \Downarrow \sigma \qquad \textsc{Ass}_{\text{BS}}\colon \langle x\ \texttt{:=}\ a,\ \sigma\rangle \Downarrow \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]$$

$$\textsc{Seq}_{\text{BS}}\colon \frac{\langle c_0,\ \sigma\rangle \Downarrow \sigma' \qquad \langle c_1,\ \sigma'\rangle \Downarrow \sigma''}{\langle c_0\texttt{;}\ c_1,\ \sigma\rangle \Downarrow \sigma''}$$

$$\textsc{IfTT}_{\text{BS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c_0,\ \sigma\rangle \Downarrow \sigma'}{\langle \texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\ \sigma\rangle \Downarrow \sigma'} \qquad \textsc{IfFF}_{\text{BS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \qquad \langle c_1,\ \sigma\rangle \Downarrow \sigma'}{\langle \texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\ \sigma\rangle \Downarrow \sigma'}$$

$$\textsc{WhileFF}_{\text{BS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}}{\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \Downarrow \sigma}$$

$$\textsc{WhileTT}_{\text{BS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c,\ \sigma\rangle \Downarrow \sigma' \qquad \langle \texttt{while (}b\texttt{) do }c,\ \sigma'\rangle \Downarrow \sigma''}{\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \Downarrow \sigma''}$$

Formal ist $\langle \_,\ \_\rangle \Downarrow \_$ die kleinste Menge über $\mathsf{Com} \times \Sigma \times \Sigma$, die unter obigen Regeln abgeschlossen ist. Damit ergibt sich auch folgende Induktionsregel für $\langle \_,\ \_\rangle \Downarrow \_$:

$$\frac{\begin{array}{c}\langle c,\ \sigma\rangle \Downarrow \sigma' \qquad \forall\sigma.\ P(\texttt{skip},\sigma,\sigma) \qquad \forall x,a,\sigma.\ P(x\ \texttt{:=}\ a,\sigma,\sigma[x\mapsto\mathcal{A}[\![a]\!]\,\sigma])\\ \forall c_0,c_1,\sigma,\sigma',\sigma''.\ \langle c_0,\ \sigma\rangle \Downarrow \sigma' \wedge \langle c_1,\ \sigma'\rangle \Downarrow \sigma'' \wedge P(c_0,\sigma,\sigma') \wedge P(c_1,\sigma',\sigma'') \longrightarrow P(c_0\texttt{;}\ c_1,\sigma,\sigma'')\\ \forall b,c_0,c_1,\sigma,\sigma'.\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \wedge \langle c_0,\ \sigma\rangle \Downarrow \sigma' \wedge P(c_0,\sigma,\sigma') \longrightarrow P(\texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\sigma,\sigma')\\ \forall b,c_0,c_1,\sigma,\sigma'.\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \wedge \langle c_1,\ \sigma\rangle \Downarrow \sigma' \wedge P(c_1,\sigma,\sigma') \longrightarrow P(\texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\sigma,\sigma')\\ \forall b,c,\sigma.\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \longrightarrow P(\texttt{while (}b\texttt{) do }c,\sigma,\sigma)\\ \forall b,c,\sigma,\sigma',\sigma''.\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \wedge \langle c,\ \sigma\rangle \Downarrow \sigma' \wedge \langle \texttt{while (}b\texttt{) do }c,\ \sigma'\rangle \Downarrow \sigma'' \wedge\\ P(c,\sigma,\sigma') \wedge P(\texttt{while (}b\texttt{) do }c,\sigma',\sigma'') \longrightarrow P(\texttt{while (}b\texttt{) do }c,\sigma,\sigma'')\end{array}}{P(c,\sigma,\sigma')}$$

Beispiel 3.
Semantik des Programms `z := x; (x := y; y := z)` im Zustand $\sigma_0 = \varepsilon[\mathtt{x} \mapsto 5, \mathtt{y} \mapsto 7, \mathtt{z} \mapsto 0]$ als Ableitungsbaum:

$$\dfrac{\dfrac{}{\langle \mathtt{z := x},\ \sigma_0\rangle \Downarrow \sigma_1}\ \textsc{Ass}_{\text{BS}} \qquad \dfrac{\dfrac{}{\langle \mathtt{x := y},\ \sigma_1\rangle \Downarrow \sigma_2}\ \textsc{Ass}_{\text{BS}} \qquad \dfrac{}{\langle \mathtt{y := z},\ \sigma_2\rangle \Downarrow \sigma_3}\ \textsc{Ass}_{\text{BS}}}{\langle \mathtt{x := y;\ y := z},\ \sigma_1\rangle \Downarrow \sigma_3}\ \textsc{Seq}_{\text{BS}}}{\langle \mathtt{z := x;\ (x := y;\ y := z)},\ \sigma_0\rangle \Downarrow \sigma_3}\ \textsc{Seq}_{\text{BS}}$$

wobei $\sigma_1 = \sigma_0[\mathtt{z} \mapsto 5]$, $\sigma_2 = \sigma_1[\mathtt{x} \mapsto 7]$ und $\sigma_3 = \sigma_2[\mathtt{y} \mapsto 5]$, also $\sigma_3 = \varepsilon[\mathtt{x} \mapsto 7, \mathtt{y} \mapsto 5, \mathtt{z} \mapsto 5]$.

Übung: Was ist der Ableitungsbaum von folgendem Programm für den Anfangszustand $\sigma_0 = \varepsilon[\mathtt{x} \mapsto 13,\ \mathtt{y} \mapsto 5,\ \mathtt{z} \mapsto 9]$?

```
z := 0; while (y <= x) do (z := z + 1; x := x - y)
```

Lemma 1 (Schleifenabwicklungslemma).
`while (`b`) do` c hat das gleiche Verhalten wie `if (`b`) then (`c`; while (`b`) do` c`) else skip`.

Beweis. Sei $w =$ `while (`b`) do` c und $w' =$ `if (`b`) then` c`; while (`b`) do` c `else skip`. Zu zeigen: $\langle w,\ \sigma\rangle \Downarrow \sigma'$ gilt genau dann, wenn $\langle w',\ \sigma\rangle \Downarrow \sigma'$. Beweis: Fallunterscheidung nach $\mathcal{B}[\![b]\!]\sigma$:

- Fall $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$: Nach den Regeln der Big-Step-Semantik lässt sich $\langle w,\ \sigma\rangle \Downarrow \sigma'$ nur mit der Regel $\textsc{WhileFF}_{\text{BS}}$ ableiten (Regelinversion); $\langle w',\ \sigma\rangle \Downarrow \sigma'$ nur mit den Regeln $\textsc{IfFF}_{\text{BS}}$ und $\textsc{Skip}_{\text{BS}}$. Also:

$$\dfrac{\mathcal{B}[\![b]\!]\sigma = \mathbf{ff} \qquad \sigma = \sigma'}{\langle w,\ \sigma\rangle \Downarrow \sigma'} \quad \Leftrightarrow \quad \dfrac{\mathcal{B}[\![b]\!]\sigma = \mathbf{ff} \qquad \dfrac{\sigma = \sigma'}{\langle \mathtt{skip},\ \sigma\rangle \Downarrow \sigma'}}{\langle w',\ \sigma\rangle \Downarrow \sigma'}$$

- Fall $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$: Wieder mit Regelinversion gibt es nur die Regel $\textsc{WhileTT}_{\text{BS}}$ für $\langle w,\ \sigma\rangle \Downarrow \sigma'$ und $\textsc{IfTT}_{\text{BS}}$ und $\textsc{Seq}_{\text{BS}}$ für $\langle w',\ \sigma\rangle \Downarrow \sigma'$. Also:

$$\dfrac{\mathcal{B}[\![b]\!]\sigma = \mathbf{tt} \qquad \dfrac{A}{\langle c,\ \sigma\rangle \Downarrow \sigma^*} \qquad \dfrac{B}{\langle w,\ \sigma^*\rangle \Downarrow \sigma'}}{\langle w,\ \sigma\rangle \Downarrow \sigma'} \quad \Leftrightarrow \quad \dfrac{\mathcal{B}[\![b]\!]\sigma = \mathbf{tt} \qquad \dfrac{\dfrac{A}{\langle c,\ \sigma\rangle \Downarrow \sigma^*} \qquad \dfrac{B}{\langle w,\ \sigma^*\rangle \Downarrow \sigma'}}{\langle \mathtt{c;}\ w,\ \sigma\rangle \Downarrow \sigma'}}{\langle w',\ \sigma\rangle \Downarrow \sigma'}$$ □

3.2 Determinismus der Big-Step-Semantik

Eine Semantik ist *deterministisch*, wenn sie jedem Programm und jedem Anfangszustand maximal ein Verhalten zuordnet. Für eine Big-Step-Semantik heißt dies konkret, dass dann auch die Endzustände gleich sind.

Theorem 2 (Determinismus). $\langle \_,\ \_\rangle \Downarrow \_$ ist deterministisch.

Beweis. Zu zeigen: Falls $\langle c,\ \sigma_0\rangle \Downarrow \sigma_1$ und $\langle c,\ \sigma_0\rangle \Downarrow \sigma_2$, dann gilt $\sigma_1 = \sigma_2$.
Beweis: Induktion nach $\langle c,\ \sigma_0\rangle \Downarrow \sigma_1$ (σ_2 beliebig). Damit $P(c, \sigma_0, \sigma_1) \equiv \forall \sigma_2.\ \langle c,\ \sigma_0\rangle \Downarrow \sigma_2 \longrightarrow \sigma_1 = \sigma_2$.

- Fall $\textsc{Skip}_{\text{BS}}$: Zu zeigen: Für alle σ gilt $P(\mathtt{skip}, \sigma, \sigma)$, d.h. $\forall \sigma_2.\ \langle \mathtt{skip},\ \sigma\rangle \Downarrow \sigma_2 \longrightarrow \sigma = \sigma_2$.
  Sei also σ_2 beliebig mit $\langle \mathtt{skip},\ \sigma\rangle \Downarrow \sigma_2$. Aus den Regeln der Big-Step-Semantik lässt sich dies nur mit der Regel $\textsc{Skip}_{\text{BS}}$ ableiten (Regelinversion). Damit folgt $\sigma_2 = \sigma$.

- Fall Ass_{BS}: Zu zeigen: Für alle x, a und σ gilt $P(x \texttt{ := } a, \sigma, \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma])$, d.h.

$$\forall \sigma_2.\ \langle x \texttt{ := } a,\ \sigma\rangle \Downarrow \sigma_2 \longrightarrow \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma] = \sigma_2$$

  Sei also σ_2 beliebig mit $\langle x \texttt{ := } a,\ \sigma\rangle \Downarrow \sigma_2$. Durch Regelinversion ($\text{Ass}_{\text{BS}}$) folgt $\sigma_2 = \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]$.

- Fall Seq_{BS}: Zu zeigen: Für alle c_0, c_1, σ, σ' und σ'' mit $\langle c_0,\ \sigma\rangle \Downarrow \sigma'$ und $\langle c_1,\ \sigma'\rangle \Downarrow \sigma''$ gilt: Aus $P(c_0, \sigma, \sigma')$ und $P(c_1, \sigma', \sigma'')$ folgt $P(c_0\texttt{;}\ c_1, \sigma, \sigma'')$, d.h.:

$$\begin{aligned}&\big(\forall \sigma_2.\ \langle c_0,\ \sigma\rangle \Downarrow \sigma_2 \longrightarrow \sigma' = \sigma_2\big) \wedge \big(\forall \sigma_2.\ \langle c_1,\ \sigma'\rangle \Downarrow \sigma_2 \longrightarrow \sigma'' = \sigma_2\big)\\ &\longrightarrow \big(\forall \sigma_2.\ \langle c_0\texttt{;}\ c_1,\ \sigma\rangle \Downarrow \sigma_2 \longrightarrow \sigma'' = \sigma_2\big)\end{aligned}$$

  Sei also σ_2 beliebig mit $\langle c_0\texttt{;}\ c_1,\ \sigma\rangle \Downarrow \sigma_2$. Mit Regelinversion ($\text{Seq}_{\text{BS}}$) gibt es ein σ^*, so dass $\langle c_0,\ \sigma\rangle \Downarrow \sigma^*$ und $\langle c_1,\ \sigma^*\rangle \Downarrow \sigma_2$. Nach Induktionsannahme $P(c_0, \sigma, \sigma')$ folgt aus $\langle c_0,\ \sigma\rangle \Downarrow \sigma^*$, dass $\sigma' = \sigma^*$. Damit gilt auch $\langle c_1,\ \sigma'\rangle \Downarrow \sigma_2$ und mit der Induktionsannahme $P(c_1, \sigma', \sigma'')$ folgt die Behauptung $\sigma'' = \sigma_2$.

- Fall IfTT_{BS}: Zu zeigen: Für alle b, c_0, c_1, σ und σ' mit $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$ und $\langle c_0,\ \sigma\rangle \Downarrow \sigma'$ gilt: Aus $P(c_0, \sigma, \sigma')$ folgt $P(\texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1, \sigma, \sigma')$.
  Sei also σ_2 beliebig mit $\langle \texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\ \sigma\rangle \Downarrow \sigma_2$, was nur durch die Regeln IfTT_{BS} und IfFF_{BS} ableitbar sein könnte. Wegen $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$ ist IfFF_{BS} ausgeschlossen. Damit folgt dass $\langle c_0,\ \sigma\rangle \Downarrow \sigma_2$ und mit der Induktionsannahme $P(c_0, \sigma, \sigma')$ die Behauptung $\sigma' = \sigma_2$.

- Fall IfFF_{BS}: Analog zu IfTT_{BS}.

- Fall $\text{WhileTT}_{\text{BS}}$:
  Zu zeigen: Für alle b, c, σ, σ' und σ'' mit $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$, $\langle c,\ \sigma\rangle \Downarrow \sigma'$ und $\langle \texttt{while (}b\texttt{) do }c,\ \sigma'\rangle \Downarrow \sigma''$ gilt: Aus $P(c, \sigma, \sigma')$ und $P(\texttt{while (}b\texttt{) do }c, \sigma', \sigma'')$ folgt $P(\texttt{while (}b\texttt{) do }c, \sigma, \sigma'')$.
  Sei also σ_2 beliebig mit $\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \Downarrow \sigma_2$. Mit Regelinversion ($\text{WhileTT}_{\text{BS}}$, $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$ schließt $\text{WhileFF}_{\text{BS}}$ aus) gibt es ein σ^*, so dass $\langle c,\ \sigma\rangle \Downarrow \sigma^*$ und $\langle \texttt{while (}b\texttt{) do }c,\ \sigma^*\rangle \Downarrow \sigma_2$. Aus $\langle c,\ \sigma\rangle \Downarrow \sigma^*$ folgt mit der Induktionsannahme $P(c, \sigma, \sigma')$, dass $\sigma' = \sigma^*$. Damit folgt mit der Induktionsannahme $P(\texttt{while (}b\texttt{) do }c, \sigma', \sigma'')$ aus $\langle \texttt{while (}b\texttt{) do }c,\ \sigma^*\rangle \Downarrow \sigma_2$ die Behauptung, dass $\sigma'' = \sigma_2$.

- Fall $\text{WhileFF}_{\text{BS}}$: Zu zeigen: Für alle b, c und σ mit $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$ gilt $P(\texttt{while (}b\texttt{) do }c, \sigma, \sigma)$.
  Sei also σ_2 beliebig mit $\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \Downarrow \sigma''$. Nach Regelinversion ($\text{WhileFF}_{\text{BS}}$, $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$ schließt $\text{WhileTT}_{\text{BS}}$ aus) folgt die Behauptung $\sigma = \sigma_2$. □

3.3 Small-Step-Semantik für While

Kern einer Small-Step-Semantik ist eine Ein-Schritt-Auswertungsrelation $\langle c,\ \sigma\rangle \to_1 \langle c',\ \sigma'\rangle$, die für ein Programm c und Zustand σ einen einzelnen Rechenschritt beschreibt: c' ist der Rest des Programms, der noch im neuen Zustand σ' auszuführen verbleibt.

Definition 5 (Small-Step-Semantik für While). Die Ein-Schritt-Auswertungsrelation $\to_1$ der Small-Step-Semantik ist induktiv über den Syntaxbaum definiert durch

$$\text{Ass}_{\text{SS}}\colon \langle x \texttt{ := } a,\ \sigma\rangle \to_1 \langle \texttt{skip},\ \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]\rangle$$

$$\text{Seq1}_{\text{SS}}\colon \frac{\langle c_0,\ \sigma\rangle \to_1 \langle c_0',\ \sigma'\rangle}{\langle c_0\texttt{;}\ c_1,\ \sigma\rangle \to_1 \langle c_0'\texttt{;}\ c_1,\ \sigma'\rangle} \qquad \text{Seq2}_{\text{SS}}\colon \langle \texttt{skip;}\ c,\ \sigma\rangle \to_1 \langle c,\ \sigma\rangle$$

$$\text{IfTT}_{\text{SS}}\colon \frac{\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}}{\langle \texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\ \sigma\rangle \to_1 \langle c_0,\ \sigma\rangle} \qquad \text{IfFF}_{\text{SS}}\colon \frac{\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}}{\langle \texttt{if (}b\texttt{) then }c_0\texttt{ else }c_1,\ \sigma\rangle \to_1 \langle c_1,\ \sigma\rangle}$$

$$\text{While}_{\text{SS}}\colon \langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \to_1 \langle \texttt{if (}b\texttt{) then }c\texttt{; while (}b\texttt{) do }c\texttt{ else skip},\ \sigma\rangle$$

Definition 6 (blockiert).
Eine Konfiguration, die nicht weiter auswerten kann, ist *blockiert*, notiert als $\langle c,\ \sigma\rangle \not\to_1$.

Für die Anweisung `skip` gibt es keine Auswertungsregel: $\langle \texttt{skip},\ \sigma\rangle$ bezeichnet eine Endkonfiguration des Programms, σ ist der Endzustand. Kennzeichen einer guten Semantik ist, dass von allen (wohlgeformten) Konfigurationen nur Endkonfigurationen blockiert sind.

Definition 7 (Ableitungsfolge). Eine *Ableitungsfolge* für $\gamma_0 = \langle c,\ \sigma\rangle$ ist eine (endliche oder unendliche) Folge $(\gamma_i)_i$ mit $\gamma_0 \to_1 \gamma_1 \to_1 \gamma_2 \to_1 \ldots$. Sie ist *maximal*, falls $(\gamma_i)_i$ unendlich ist oder das letzte γ_k keine Reduktion in $\to_1$ besitzt. $\gamma \overset{n}{\to}_1 \gamma'$ ($\gamma \overset{*}{\to}_1 \gamma'$) bezeichne, dass es eine Ableitungsfolge mit n (endlich vielen) Schritten von γ nach γ' gibt. $\overset{*}{\to}_1$ ist die reflexive, transitive Hülle von $\to_1$.

Maximale Ableitungsfolgen beschreiben das Programmverhalten. Nichtterminierende Ausführungen entsprechen unendlichen Ableitungsfolgen – diese haben *keinen* Endzustand; $\gamma \overset{\infty}{\to}_1$ bezeichne, dass es eine unendlich lange Ableitungsfolge gibt, die in γ beginnt.

Beispiel 4. Semantik des Programms `z := x; (x := y; y := z)` im Zustand $\sigma_0 = \varepsilon[\mathtt{x}\mapsto 5, \mathtt{y}\mapsto 7, \mathtt{z}\mapsto 0]$ als maximale Ableitungsfolge:

$$\begin{aligned}&\langle \texttt{z := x; (x := y; y := z)},\ \sigma_0\rangle \to_1 \langle \texttt{skip; (x := y; y := z)},\ \sigma_1\rangle\\ \to_1&\langle \texttt{x := y; y := z},\ \sigma_1\rangle \to_1 \langle \texttt{skip; y := z},\ \sigma_2\rangle \to_1 \langle \texttt{y := z},\ \sigma_2\rangle \to_1 \langle \texttt{skip},\ \sigma_3\rangle\end{aligned}$$

wobei $\sigma_1 = \sigma_0[\mathtt{z}\mapsto 5]$, $\sigma_2 = \sigma_1[\mathtt{x}\mapsto 7]$ und $\sigma_3 = \sigma_2[\mathtt{y}\mapsto 5]$, also $\sigma_3 = \varepsilon[\mathtt{x}\mapsto 7, \mathtt{y}\mapsto 5, \mathtt{z}\mapsto 5]$. Jeder einzelne Schritt muss dabei durch einen Ableitungsbaum für $\to_1$ gerechtfertigt werden, z. B.:

$$\dfrac{\dfrac{}{\langle \texttt{z := x},\ \sigma_0\rangle \to_1 \langle \texttt{skip},\ \sigma_1\rangle}\ \textsc{Ass}_{\textsc{ss}}}{\langle \texttt{z := x; (x := y; y := z)},\ \sigma_0\rangle \to_1 \langle \texttt{skip; (x := y; y := z)},\ \sigma_1\rangle}\ \textsc{Seq1}_{\textsc{ss}}$$

Beispiel 5 (Nichttermination).
Sei $w = \texttt{while (not (x == 1)) do x := x + 1}$ und $\sigma_n = [\mathtt{x}\mapsto n]$. Für $\langle w,\ \sigma_0\rangle$ ergibt sich folgende maximale (endiche) Ableitungsfolge:

$$\begin{aligned}\langle w,\ \sigma_0\rangle \to_1 &\overbrace{\langle \texttt{if (not (x == 1)) then x := x + 1; } w \texttt{ else skip},\ \sigma_0\rangle}^{=\text{if}}\\ &\to_1 \langle \texttt{x := x + 1; } w,\ \sigma_0\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_1\rangle \to_1 \langle w,\ \sigma_1\rangle \to_1 \langle \text{if},\ \sigma_1\rangle \to_1 \langle \texttt{skip},\ \sigma_1\rangle\end{aligned}$$

Für $\langle w,\ \sigma_2\rangle$ ist die maximale Ableitungsfolge[1] unendlich:

$$\begin{aligned}&\langle w,\ \sigma_2\rangle \to_1 \langle \text{if},\ \sigma_2\rangle \to_1 \langle \texttt{x := x + 1; } w,\ \sigma_2\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_3\rangle\\ \to_1&\langle w,\ \sigma_3\rangle \to_1 \langle \text{if},\ \sigma_3\rangle \to_1 \langle \texttt{x := x + 1; } w,\ \sigma_3\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_4\rangle\\ \to_1&\langle w,\ \sigma_4\rangle \to_1 \ldots\end{aligned}$$

Häufig beschreiben die einzelnen Schritte maximaler Ableitungsfolgen zu detailliert, was ein Programm berechnet. Beispielsweise haben die Programm `skip; skip` und `skip` unterschiedliche maximale Ableitungsfolgen und damit eine unterschiedliche Semantik. Deswegen abstrahiert man üblicherweise von den maximalen Ableitungsfolgen und nimmt die transitive Hülle $\overset{*}{\to}_1$ zu einer Endkonfiguration beziehungsweise $\overset{\infty}{\to}_1$ als Semantik eines Programms.

[1] Für While sind maximale Ableitungsfolgen eindeutig, s. Kor. 5 unten

Formal gesehen sind $\overset{*}{\to}_1$ und $\overset{n}{\to}_1$ ganz unterschiedlich definiert. Es gilt aber

$$\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle c',\ \sigma'\rangle \quad \text{gdw.} \quad \exists n.\ \langle c,\ \sigma\rangle \overset{n}{\to}_1 \langle c',\ \sigma'\rangle$$

Deswegen werden wir im Folgenden bei Bedarf von der $\overset{*}{\to}_1$ auf $\overset{n}{\to}_1$ und wieder zurück wechseln – unter Beachtung, dass n existenziell quantifiziert ist.

Lemma 3 (Fortschritt). `skip` ist das einzige Programm, das blockiert ist.

Beweis. Zu zeigen: Für alle Programme c außer `skip` und jeden Zustand σ gibt es c' und σ' mit $\langle c,\ \sigma\rangle \to_1 \langle c',\ \sigma'\rangle$. Beweis mittels Induktion über c:

- Fall $c =$ `skip`: Explizit ausgeschlossen.
- Fälle $c =$ `x := a`, $c =$ `if (b) then` c_1 `else` c_2 und $c =$ `while (b) do` c_0:
  Folgen direkt aus den Regeln Ass_{SS}, IfTT_{SS}, IfFF_{SS} (Fallunterscheidung nach $\mathcal{B}\,[\![b]\!]\,\sigma$) und While_{SS}.
- Fall $c = c_1$`;` c_2: Induktionshypothesen:

  (i) Falls $c_1 \neq$ `skip`, dann gibt es c_1' und σ_1' mit $\langle c_1,\ \sigma\rangle \to_1 \langle c_1',\ \sigma_1'\rangle$.

  (ii) Falls $c_2 \neq$ `skip`, dann gibt es c_2' und σ_2' mit $\langle c_2,\ \sigma\rangle \to_1 \langle c_2',\ \sigma_2'\rangle$.

  Zu zeigen: Es gibt c' und σ' mit $\langle c_1$`;` $c_2,\ \sigma\rangle \to_1 \langle c',\ \sigma'\rangle$.
  Fallunterscheidung nach $c_1 =$ `skip`:
  – Fall $c_1 =$ `skip`: Folgt direkt aus Regel Seq2_{SS}
  – Fall $c_1 \neq$ `skip`: Mit Induktionshypothese (i) gibt es c_1' und σ_1' mit $\langle c_1,\ \sigma\rangle \to_1 \langle c_1',\ \sigma_1'\rangle$. Mit Regel Seq1_{SS} folgt $\langle c_1$`;` $c_2,\ \sigma\rangle \to_1 \langle c_1'$`;` $c_2,\ \sigma_1'\rangle$. □

Im Progress-Beweis wurden alle Regeln für $\to_1$ benutzt, keine ist also überflüssig.

Theorem 4 (Determinismus). $\langle \_,\ \_\rangle \to_1 \langle \_,\ \_\rangle$ ist deterministisch.

Beweis analog zum Determinismus für die Big-Step-Semantik $\langle \_,\ \_\rangle \Downarrow \_$ (Thm. 2).

Korollar 5. Für alle c und σ gibt es genau eine maximale Ableitungsfolge.

Die Existenz einer maximalen Ableitungsfolge folgt aus der Existenz unendlich langer Folgen, die Eindeutigkeit aus dem Determinismus durch Induktion.

3.4 Äquivalenz zwischen Big-Step- und Small-Step-Semantik

Big-Step- und Small-Step-Semantik sind zwei Semantik-Definitionen für While. Bei der Big-Step-Semantik interessiert nur der Endzustand einer Programmausführung, während eine Ableitungsfolge in der Small-Step-Semantik zusätzlich alle einzelnen Zwischenberechnungsschritte und -zustände enthält. Trotzdem passen beide Definitionen wie folgt zusammen, was in diesem Abschnitt bewiesen wird:

$$\langle c,\ \sigma\rangle \Downarrow \sigma' \text{ genau dann, wenn } \langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$$

Die Äquivalenzbeweise benötigen die beiden folgenden Lemmas für Sequenz. Gäbe es mehr Sprachkonstrukte mit einer rekursiven Regel für die Small-Step-Semantik wie Seq1_{SS}, bräuchte man entsprechende Lemmas für jedes dieser.

Lemma 6 (Liftinglemma für Sequenz). Falls $\langle c,\ \sigma\rangle \overset{n}{\to}_1 \langle c',\ \sigma'\rangle$, dann $\langle c$`;` $c_2,\ \sigma\rangle \overset{n}{\to}_1 \langle c'$`;` $c_2,\ \sigma'\rangle$.

Beweis. Induktion über n, der Induktionsschritt folgt aus der Regel Seq1_{SS}. □

Lemma 7 (Zerlegungslemma für Sequenz). Wenn $\langle c_1\,;\ c_2,\ \sigma\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$, dann gibt es i, j und σ', so dass $\langle c_1,\ \sigma\rangle \overset{i}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ und $\langle c_2,\ \sigma'\rangle \overset{j}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ mit $i + j + 1 = n$.

Beweis. Beweis per Induktion über n (c_1 und σ beliebig):

- Basisfall $n = 0$: Dieser Fall ist unmöglich, weil $c_1\,;\ c_2 \neq \texttt{skip}$.
- Induktionsschritt $n + 1$: Induktionsannahme: Für alle c_1 und σ gilt: Wenn $\langle c_1\,;\ c_2,\ \sigma\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$, dann gibt es i, j und σ' mit $\langle c_1,\ \sigma\rangle \overset{i}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$, $\langle c_2,\ \sigma'\rangle \overset{j}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ und $i + j + 1 = n$.

  Unter der Annahme $\langle c_1\,;\ c_2,\ \sigma\rangle \overset{n+1}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ ist zu zeigen, dass es i, j und σ' gibt mit $\langle c_1,\ \sigma\rangle \overset{i}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$, $\langle c_2,\ \sigma'\rangle \overset{j}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ und $i + j + 1 = n + 1$.

  Beweis: Wegen $\langle c_1\,;\ c_2,\ \sigma\rangle \overset{n+1}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ gibt es ein c und σ^*, so dass

  $$\langle c_1\,;\ c_2,\ \sigma\rangle \to_1 \langle c,\ \sigma^*\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle.$$

  Mit Regelinversion folgt aus $\langle c_1\,;\ c_2,\ \sigma\rangle \to_1 \langle c,\ \sigma^*\rangle$, dass entweder ($\textsc{Seq2}_{\text{SS}}$) $c_1 = \texttt{skip}$, $c = c_2$, $\sigma^* = \sigma$ oder ($\textsc{Seq1}_{\text{SS}}$) c von der Form $c_1'\,;\ c_2$ mit $\langle c_1,\ \sigma\rangle \to_1 \langle c_1',\ \sigma^*\rangle$ ist. Im ersten Fall folgt die Behauptung mit der Aufteilung $i = 0$, $j = n$ und $\sigma' = \sigma$. Im anderen Fall ergibt die Induktionsannahme für $\langle c_1'\,;\ c_2,\ \sigma^*\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ eine Aufteilung in $\langle c_1',\ \sigma^*\rangle \overset{i'}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ und $\langle c_2,\ \sigma'\rangle \overset{j'}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$ mit $i' + j' + 1 = n$. Mit $\langle c_1,\ \sigma\rangle \to_1 \langle c_1',\ \sigma^*\rangle$ ergibt sich dann die Behauptung für $i = i' + 1$, $j = j'$ und $\sigma' = \sigma'$. □

Theorem 8 (Small-Step simuliert Big-Step). Aus $\langle c,\ \sigma\rangle \Downarrow \sigma'$ folgt $\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$.

Beweis in der Übung.

Theorem 9 (Big-Step simuliert Small-Step). Aus $\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ folgt $\langle c,\ \sigma\rangle \Downarrow \sigma'$.

Beweis. Wegen $\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ gibt es ein n mit $\langle c,\ \sigma\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$. Beweis von $\langle c,\ \sigma\rangle \Downarrow \sigma'$ per *vollständiger Induktion* über n (c, σ, σ' beliebig):

Sei n beliebig. Induktionsannahme: Für alle $m < n$ und c, σ, σ' gilt: Wenn $\langle c,\ \sigma\rangle \overset{m}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$, dann auch $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Zu zeigen: Aus (i) $\langle c,\ \sigma\rangle \overset{n}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ folgt $\langle c,\ \sigma\rangle \Downarrow \sigma'$ für beliebige c, σ und σ'.

Fallunterscheidung nach c:

- Fall $c = \texttt{skip}$: Mit (i) folgt, dass $n = 0$ und $\sigma' = \sigma$. $\langle \texttt{skip},\ \sigma\rangle \Downarrow \sigma$ folgt aus Regel $\textsc{Skip}_{\text{BS}}$.
- Fall $c = \texttt{x := } a$: Mit (i) folgt, dass $n = 1$ und $\sigma' = \sigma[\texttt{x} \mapsto \mathcal{A}\,[\![a]\!]\,\sigma]$. $\langle \texttt{x := } a,\ \sigma\rangle \Downarrow \sigma[\texttt{x} \mapsto \mathcal{A}\,[\![a]\!]\,\sigma]$ folgt aus der Regel $\textsc{Ass}_{\text{BS}}$.
- Fall $c = c_1\,;\ c_2$:

  Nach dem Zerlegungslemma lässt sich (i) in $\langle c_1,\ \sigma\rangle \overset{i}{\to}_1 \langle \texttt{skip},\ \sigma^*\rangle$ und $\langle c_2,\ \sigma^*\rangle \overset{j}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ mit $i + j + 1 = n$ aufteilen. Damit ist insbesondere $i < n$ und $j < n$, d.h., die Induktionsannahme lässt sich auf beide Teile anwenden: $\langle c_1,\ \sigma\rangle \Downarrow \sigma^*$ und $\langle c_2,\ \sigma^*\rangle \Downarrow \sigma'$. Daraus folgt die Behauptung mit der Regel $\textsc{Seq}_{\text{BS}}$.
- Fall $c = \texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2$: Aus (i) folgt mit Regelinversion, dass $n > 0$ und entweder ($\textsc{IfTT}_{\text{SS}}$) $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt}$ und $\langle c_1,\ \sigma\rangle \overset{n-1}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ oder ($\textsc{IfFF}_{\text{SS}}$) $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff}$ und $\langle c_2,\ \sigma\rangle \overset{n-1}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$. In beiden Fällen lässt sich die Induktionsannahme anwenden und die Behauptung folgt aus den Regeln $\textsc{IfTT}_{\text{BS}}$ bzw. $\textsc{IfFF}_{\text{BS}}$.
- Fall $c = \texttt{while (}b\texttt{) do } c$: Aus (i) folgt mit Regelinversion ($\textsc{While}_{\text{SS}}$), dass $n > 0$ und

  $$\langle \texttt{while (}b\texttt{) do } c,\ \sigma\rangle \to_1 \langle \underbrace{\texttt{if (}b\texttt{) then } c\texttt{; while (}b\texttt{) do } c \texttt{ else skip}}_{=w'},\ \sigma\rangle \overset{n-1}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle.$$

Wendet man die Induktionshypothese mit $m = n - 1 < n$ und $c = w'$ an, so folgt $\langle w',\ \sigma\rangle \Downarrow \sigma'$. Da w' nach dem Schleifenabwicklungslemma (Lem. 1) in der Big-Step-Semantik äquivalent zu `while (`b`) do` c ist, gilt auch $\langle$`while (`b`) do` $c,\ \sigma\rangle \Downarrow \sigma'$. □

Korollar 10 (Äquivalenz von Big-Step- und Small-Step-Semantik).
Für alle c, σ und σ' gilt $\langle c,\ \sigma\rangle \Downarrow \sigma'$ genau dann, wenn $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle$`skip`$,\ \sigma'\rangle$ gilt.

3.5 Äquivalenz von Programmen

Zu entscheiden. ob zwei Programme äquivalent sind, ist ein wesentlicher Anwendungsbereich für Semantiken. Die bisherigen Äquivalenzbegriffe sind dafür aber i. d. R. zu feingranular, wie folgendes Beispiel zeigt:

```
tmp := y; y := x; x := tmp        x := x - y; y := y + x; x := y - x
```

Beide Programme vertauschen die Inhalte von `x` und `y`, aber das erste verwendet dazu die Hilfsvariable `tmp`. Demnach sind beide Programme nicht äquivalent, weil das eine `tmp` möglicherweise verändert, das andere aber nicht. Wird im weiteren Programmverlauf der in `tmp` gespeicherte Wert aber nicht mehr verwendet, wäre es gerechtfertigt, beide Programme als äquivalent zu betrachten.

Definition 8 (Äquivalenz von Programmen). Zwei Programme c_1 und c_2 sind äquivalent bezüglich der Variablen $V \subseteq \mathsf{Var}$, falls für alle σ gilt:

- Wenn $\langle c_1,\ \sigma\rangle \Downarrow \sigma_1$ für ein σ_1, dann gibt es ein σ_2 mit $\langle c_2,\ \sigma\rangle \Downarrow \sigma_2$ und $\sigma_1(x) = \sigma_2(x)$ für alle $x \in V$.
- Wenn $\langle c_2,\ \sigma\rangle \Downarrow \sigma_2$ für ein σ_2, dann gibt es ein σ_1 mit $\langle c_1,\ \sigma\rangle \Downarrow \sigma_1$ und $\sigma_1(x) = \sigma_2(x)$ für alle $x \in V$.

In obigem Beispiel sind beide Programme äquivalent bezüglich der Variablen $\{$ `x`, `y` $\}$.

Die beiden Bedingungen sind klassische Simulationsbedingungen in beide Richtungen, man kann sie auch für Small-Step-Semantiken entsprechend formulieren. Da die Big-Step-Semantik deterministisch ist, lässt sie wie folgt vereinfachen.

Lemma 11 (Äquivalenz für deterministische Programme). Zwei Programme c_1 und c_2 sind äquivalent bezüglich V genau dann, wenn

(i) c_1 terminiert genau dann, wenn c_2 terminiert, d.h., es gibt ein σ_1 mit $\langle c_1,\ \sigma\rangle \Downarrow \sigma_1$ genau dann, wenn es ein σ_2 mit $\langle c_2,\ \sigma\rangle \Downarrow \sigma_2$ gibt.

(ii) Wenn $\langle c_1,\ \sigma\rangle \Downarrow \sigma_1$ und $\langle c_2,\ \sigma\rangle \Downarrow \sigma_2$, dann $\sigma_1(x) = \sigma_2(x)$ für alle $x \in V$.