

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:

$$\begin{array}{c}
\text{SKIP}_{\text{BS}}: \langle \text{skip}, \sigma \rangle \Downarrow \sigma \quad \text{ASS}_{\text{BS}}: \langle x := a, \sigma \rangle \Downarrow \sigma[x \mapsto \mathcal{A}[[a]] \sigma] \\
\\
\text{SEQ}_{\text{BS}}: \frac{\langle c_0, \sigma \rangle \Downarrow \sigma' \quad \langle c_1, \sigma' \rangle \Downarrow \sigma''}{\langle c_0; c_1, \sigma \rangle \Downarrow \sigma''} \\
\\
\text{IFTT}_{\text{BS}}: \frac{\mathcal{B}[[b]] \sigma = \text{tt} \quad \langle c_0, \sigma \rangle \Downarrow \sigma'}{\langle \text{if } (b) \text{ then } c_0 \text{ else } c_1, \sigma \rangle \Downarrow \sigma'} \quad \text{IFFF}_{\text{BS}}: \frac{\mathcal{B}[[b]] \sigma = \text{ff} \quad \langle c_1, \sigma \rangle \Downarrow \sigma'}{\langle \text{if } (b) \text{ then } c_0 \text{ else } c_1, \sigma \rangle \Downarrow \sigma'} \\
\\
\text{WHILEFF}_{\text{BS}}: \frac{\mathcal{B}[[b]] \sigma = \text{ff}}{\langle \text{while } (b) \text{ do } c, \sigma \rangle \Downarrow \sigma} \\
\\
\text{WHILETT}_{\text{BS}}: \frac{\mathcal{B}[[b]] \sigma = \text{tt} \quad \langle c, \sigma \rangle \Downarrow \sigma' \quad \langle \text{while } (b) \text{ do } c, \sigma' \rangle \Downarrow \sigma''}{\langle \text{while } (b) \text{ do } c, \sigma \rangle \Downarrow \sigma''}
\end{array}$$

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

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

Beispiel 3.

Semantik des Programms $z := x; (x := y; y := z)$ im Zustand $\sigma_0 = \varepsilon[x \mapsto 5, y \mapsto 7, z \mapsto 0]$ als Ableitungsbaum:

$$\frac{\frac{\frac{}{\langle z := x, \sigma_0 \rangle \Downarrow \sigma_1} \text{ASSBS}}{\frac{\frac{\frac{}{\langle x := y, \sigma_1 \rangle \Downarrow \sigma_2} \text{ASSBS} \quad \frac{\frac{}{\langle y := z, \sigma_2 \rangle \Downarrow \sigma_3} \text{ASSBS}}{\langle x := y; y := z, \sigma_1 \rangle \Downarrow \sigma_3} \text{SEQBS}}{\langle z := x; (x := y; y := z), \sigma_0 \rangle \Downarrow \sigma_3} \text{SEQBS}}{\langle z := x; (x := y; y := z), \sigma_0 \rangle \Downarrow \sigma_3} \text{SEQBS}}$$

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

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

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

Lemma 1 (Schleifenabwicklungslemma).

$\text{while } (b) \text{ do } c$ hat das gleiche Verhalten wie $\text{if } (b) \text{ then } (c; \text{while } (b) \text{ do } c) \text{ else skip}$.

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

- Fall $\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{ff}$: Nach den Regeln der Big-Step-Semantik lässt sich $\langle w, \sigma \rangle \Downarrow \sigma'$ nur mit der Regel $\text{WHILEFF}_{\text{BS}}$ ableiten (Regelinversion); $\langle w', \sigma \rangle \Downarrow \sigma'$ nur mit den Regeln IFF_{BS} und SKIP_{BS} . Also:

$$\frac{\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{ff} \quad \sigma = \sigma'}{\langle w, \sigma \rangle \Downarrow \sigma'} \Leftrightarrow \frac{\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{ff} \quad \frac{\sigma = \sigma'}{\langle \text{skip}, \sigma \rangle \Downarrow \sigma'}}{\langle w', \sigma \rangle \Downarrow \sigma'}$$

- Fall $\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{tt}$: Wieder mit Regelinversion gibt es nur die Regel $\text{WHILETT}_{\text{BS}}$ für $\langle w, \sigma \rangle \Downarrow \sigma'$ und IFTT_{BS} und SEQ_{BS} für $\langle w', \sigma \rangle \Downarrow \sigma'$. Also:

$$\frac{\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{tt} \quad \frac{A}{\langle c, \sigma \rangle \Downarrow \sigma^*} \quad \frac{B}{\langle w, \sigma^* \rangle \Downarrow \sigma'}}{\langle w, \sigma \rangle \Downarrow \sigma'} \Leftrightarrow \frac{\mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{tt} \quad \frac{\frac{A}{\langle c, \sigma \rangle \Downarrow \sigma^*} \quad \frac{B}{\langle w, \sigma^* \rangle \Downarrow \sigma'}}{\langle c; w, \sigma \rangle \Downarrow \sigma'}}{\langle w', \sigma \rangle \Downarrow \sigma'} \quad \square$$

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

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

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

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

- Fall SEQ_{BS}: Zu zeigen: Für alle $c_0, c_1, \sigma, \sigma'$ 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 $P(c_0; c_1, \sigma, \sigma'')$, d.h.:

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

Sei also σ_2 beliebig mit $\langle c_0; c_1, \sigma \rangle \Downarrow \sigma_2$. Mit Regelinversion (SEQ_{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 IF_{TT}_{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 $P(\mathbf{if} (b) \mathbf{then} c_0 \mathbf{else} c_1, \sigma, \sigma')$.

Sei also σ_2 beliebig mit $\langle \mathbf{if} (b) \mathbf{then} c_0 \mathbf{else} c_1, \sigma \rangle \Downarrow \sigma_2$, was nur durch die Regeln IF_{TT}_{BS} und IFF_{BS} ableitbar sein könnte. Wegen $\mathcal{B}[[b]]\sigma = \mathbf{tt}$ ist IFF_{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 IFF_{BS}: Analog zu IF_{TT}_{BS}.

- Fall WHILE_{TT}_{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 $\langle \mathbf{while} (b) \mathbf{do} c, \sigma' \rangle \Downarrow \sigma''$ gilt: Aus $P(c, \sigma, \sigma')$ und $P(\mathbf{while} (b) \mathbf{do} c, \sigma', \sigma'')$ folgt $P(\mathbf{while} (b) \mathbf{do} c, \sigma, \sigma'')$.

Sei also σ_2 beliebig mit $\langle \mathbf{while} (b) \mathbf{do} c, \sigma \rangle \Downarrow \sigma_2$. Mit Regelinversion (WHILE_{TT}_{BS}, $\mathcal{B}[[b]]\sigma = \mathbf{tt}$ schließt WHILE_{FF}_{BS} aus) gibt es ein σ^* , so dass $\langle c, \sigma \rangle \Downarrow \sigma^*$ und $\langle \mathbf{while} (b) \mathbf{do} c, \sigma^* \rangle \Downarrow \sigma_2$. Aus $\langle c, \sigma \rangle \Downarrow \sigma^*$ folgt mit der Induktionsannahme $P(c, \sigma, \sigma')$, dass $\sigma' = \sigma^*$. Damit folgt mit der Induktionsannahme $P(\mathbf{while} (b) \mathbf{do} c, \sigma', \sigma'')$ aus $\langle \mathbf{while} (b) \mathbf{do} c, \sigma^* \rangle \Downarrow \sigma_2$ die Behauptung, dass $\sigma'' = \sigma_2$.

- Fall WHILE_{FF}_{BS}: Zu zeigen: Für alle b, c und σ mit $\mathcal{B}[[b]]\sigma = \mathbf{ff}$ gilt $P(\mathbf{while} (b) \mathbf{do} c, \sigma, \sigma)$.

Sei also σ_2 beliebig mit $\langle \mathbf{while} (b) \mathbf{do} c, \sigma \rangle \Downarrow \sigma_2$. Nach Regelinversion (WHILE_{FF}_{BS}, $\mathcal{B}[[b]]\sigma = \mathbf{ff}$ schließt WHILE_{TT}_{BS} aus) folgt die Behauptung $\sigma = \sigma_2$. \square

3.3 Small-Step-Semantik für While

Kern einer Small-Step-Semantik ist eine Ein-Schritt-Auswertungsrelation $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$, 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 \rightarrow_1 der Small-Step-Semantik ist induktiv über den Syntaxbaum definiert durch

$$\text{ASS}_{\text{SS}}: \langle x := a, \sigma \rangle \rightarrow_1 \langle \mathbf{skip}, \sigma[x \mapsto \mathcal{A}[[a]]\sigma] \rangle$$

$$\text{SEQ}_{1\text{SS}}: \frac{\langle c_0, \sigma \rangle \rightarrow_1 \langle c'_0, \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \rightarrow_1 \langle c'_0; c_1, \sigma' \rangle} \quad \text{SEQ}_{2\text{SS}}: \langle \mathbf{skip}; c, \sigma \rangle \rightarrow_1 \langle c, \sigma \rangle$$

$$\text{IF}_{\text{TT}}_{\text{SS}}: \frac{\mathcal{B}[[b]]\sigma = \mathbf{tt}}{\langle \mathbf{if} (b) \mathbf{then} c_0 \mathbf{else} c_1, \sigma \rangle \rightarrow_1 \langle c_0, \sigma \rangle} \quad \text{IFF}_{\text{SS}}: \frac{\mathcal{B}[[b]]\sigma = \mathbf{ff}}{\langle \mathbf{if} (b) \mathbf{then} c_0 \mathbf{else} c_1, \sigma \rangle \rightarrow_1 \langle c_1, \sigma \rangle}$$

$$\text{WHILE}_{\text{SS}}: \langle \mathbf{while} (b) \mathbf{do} c, \sigma \rangle \rightarrow_1 \langle \mathbf{if} (b) \mathbf{then} c; \mathbf{while} (b) \mathbf{do} c \mathbf{else} \mathbf{skip}, \sigma \rangle$$

Definition 6 (blockiert).

Eine Konfiguration, die nicht weiter auswerten kann, ist *blockiert*, notiert als $\langle c, \sigma \rangle \not\rightarrow_1$.

Für die Anweisung `skip` gibt es keine Auswertungsregel: $\langle \text{skip}, \sigma \rangle$ 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 $\gamma_0 \rightarrow_1 \gamma_1 \rightarrow_1 \gamma_2 \rightarrow_1 \dots$. Sie ist *maximal*, falls $(\gamma_i)_i$ unendlich ist oder das letzte γ_k keine Reduktion in \rightarrow_1 besitzt. $\gamma \xrightarrow{n}_1 \gamma'$ ($\gamma \xrightarrow{*}_1 \gamma'$) bezeichne, dass es eine Ableitungsfolge mit n (endlich vielen) Schritten von γ nach γ' gibt. $\xrightarrow{*}_1$ ist die reflexive, transitive Hülle von \rightarrow_1 .

Maximale Ableitungsfolgen beschreiben das Programmverhalten. Nichtterminierende Ausführungen entsprechen unendlichen Ableitungsfolgen – diese haben *keinen* Endzustand; $\gamma \xrightarrow{\infty}_1$ 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 $\sigma_0 = \varepsilon[x \mapsto 5, y \mapsto 7, z \mapsto 0]$ als maximale Ableitungsfolge:

$$\begin{aligned} &\langle z := x; (x := y; y := z), \sigma_0 \rangle \rightarrow_1 \langle \text{skip}; (x := y; y := z), \sigma_1 \rangle \\ &\rightarrow_1 \langle x := y; y := z, \sigma_1 \rangle \rightarrow_1 \langle \text{skip}; y := z, \sigma_2 \rangle \rightarrow_1 \langle y := z, \sigma_2 \rangle \rightarrow_1 \langle \text{skip}, \sigma_3 \rangle \end{aligned}$$

wobei $\sigma_1 = \sigma_0[z \mapsto 5]$, $\sigma_2 = \sigma_1[x \mapsto 7]$ und $\sigma_3 = \sigma_2[y \mapsto 5]$, also $\sigma_3 = \varepsilon[x \mapsto 7, y \mapsto 5, z \mapsto 5]$. Jeder einzelne Schritt muss dabei durch einen Ableitungsbaum für \rightarrow_1 gerechtfertigt werden, z. B.:

$$\frac{\frac{}{\langle z := x, \sigma_0 \rangle \rightarrow_1 \langle \text{skip}, \sigma_1 \rangle} \text{ASS}_{\text{SS}}}{\langle z := x; (x := y; y := z), \sigma_0 \rangle \rightarrow_1 \langle \text{skip}; (x := y; y := z), \sigma_1 \rangle} \text{SEQ}_{1\text{SS}}$$

Beispiel 5 (Nichttermination).

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

$$\begin{aligned} &\langle w, \sigma_0 \rangle \rightarrow_1 \overbrace{\langle \text{if } (\text{not } (x == 1)) \text{ then } x := x + 1; w \text{ else skip}, \sigma_0 \rangle}^{\text{=if}} \\ &\rightarrow_1 \langle x := x + 1; w, \sigma_0 \rangle \rightarrow_1 \langle \text{skip}; w, \sigma_1 \rangle \rightarrow_1 \langle w, \sigma_1 \rangle \rightarrow_1 \langle \text{if}, \sigma_1 \rangle \rightarrow_1 \langle \text{skip}, \sigma_1 \rangle \end{aligned}$$

Für $\langle w, \sigma_2 \rangle$ ist die maximale Ableitungsfolge¹ unendlich:

$$\begin{aligned} &\langle w, \sigma_2 \rangle \rightarrow_1 \langle \text{if}, \sigma_2 \rangle \rightarrow_1 \langle x := x + 1; w, \sigma_2 \rangle \rightarrow_1 \langle \text{skip}; w, \sigma_3 \rangle \\ &\rightarrow_1 \langle w, \sigma_3 \rangle \rightarrow_1 \langle \text{if}, \sigma_3 \rangle \rightarrow_1 \langle x := x + 1; w, \sigma_3 \rangle \rightarrow_1 \langle \text{skip}; w, \sigma_4 \rangle \\ &\rightarrow_1 \langle w, \sigma_4 \rangle \rightarrow_1 \dots \end{aligned}$$

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 $\xrightarrow{*}_1$ zu einer Endkonfiguration beziehungsweise $\xrightarrow{\infty}_1$ als Semantik eines Programms.

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

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

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

Deswegen werden wir im Folgenden bei Bedarf von der $\xrightarrow{*}_1$ auf \xrightarrow{n}_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 $\langle c, \sigma \rangle \xrightarrow{*}_1 \langle c', \sigma' \rangle$. Beweis mittels Induktion über c :

- Fall $c = \text{skip}$: Explizit ausgeschlossen.
- Fälle $c = x := a$, $c = \text{if } (b) \text{ then } c_1 \text{ else } c_2$ und $c = \text{while } (b) \text{ do } c_0$:
Folgen direkt aus den Regeln ASS_{SS} , IFTT_{SS} , IFFF_{SS} (Fallunterscheidung nach $\mathcal{B} \llbracket b \rrbracket \sigma$) und WHILE_{SS} .
- Fall $c = c_1; c_2$: Induktionshypothesen:
 - (i) Falls $c_1 \neq \text{skip}$, dann gibt es c'_1 und σ'_1 mit $\langle c_1, \sigma \rangle \xrightarrow{*}_1 \langle c'_1, \sigma'_1 \rangle$.
 - (ii) Falls $c_2 \neq \text{skip}$, dann gibt es c'_2 und σ'_2 mit $\langle c_2, \sigma \rangle \xrightarrow{*}_1 \langle c'_2, \sigma'_2 \rangle$.

Zu zeigen: Es gibt c' und σ' mit $\langle c_1; c_2, \sigma \rangle \xrightarrow{*}_1 \langle c', \sigma' \rangle$.

Fallunterscheidung nach $c_1 = \text{skip}$:

- Fall $c_1 = \text{skip}$: Folgt direkt aus Regel SEQ2_{SS}
- Fall $c_1 \neq \text{skip}$: Mit Induktionshypothese (i) gibt es c'_1 und σ'_1 mit $\langle c_1, \sigma \rangle \xrightarrow{*}_1 \langle c'_1, \sigma'_1 \rangle$. Mit Regel SEQ1_{SS} folgt $\langle c_1; c_2, \sigma \rangle \xrightarrow{*}_1 \langle c'_1; c_2, \sigma'_1 \rangle$. □

Im Progress-Beweis wurden alle Regeln für $\xrightarrow{*}_1$ benutzt, keine ist also überflüssig.

Theorem 4 (Determinismus). $\langle -, - \rangle \xrightarrow{*}_1 \langle -, - \rangle$ 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:

$$\langle c, \sigma \rangle \Downarrow \sigma' \quad \text{genau dann, wenn} \quad \langle c, \sigma \rangle \xrightarrow{*}_1 \langle \text{skip}, \sigma' \rangle$$

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} , bräuhete man entsprechende Lemmas für jedes dieser.

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

Beweis. Induktion über n , der Induktionsschritt folgt aus der Regel SEQ1_{SS} . □

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

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

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

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

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

$$\langle c_1; c_2, \sigma \rangle \rightarrow_1 \langle c, \sigma^* \rangle \xrightarrow{n}_1 \langle \text{skip}, \sigma'' \rangle.$$

Mit Regelinversion folgt aus $\langle c_1; c_2, \sigma \rangle \rightarrow_1 \langle c, \sigma^* \rangle$, dass entweder (SEQ2_{SS}) $c_1 = \text{skip}$, $c = c_2$, $\sigma^* = \sigma$ oder (SEQ1_{SS}) c von der Form $c'_1; c_2$ mit $\langle c_1, \sigma \rangle \rightarrow_1 \langle c'_1, \sigma^* \rangle$ ist. Im ersten Fall folgt die Behauptung mit der Aufteilung $i = 0$, $j = n$ und $\sigma' = \sigma$. Im anderen Fall ergibt die Induktionsannahme für $\langle c'_1; c_2, \sigma^* \rangle \xrightarrow{n}_1 \langle \text{skip}, \sigma'' \rangle$ eine Aufteilung in $\langle c'_1, \sigma^* \rangle \xrightarrow{i'}_1 \langle \text{skip}, \sigma' \rangle$ und $\langle c_2, \sigma' \rangle \xrightarrow{j'}_1 \langle \text{skip}, \sigma'' \rangle$ mit $i' + j' + 1 = n$. Mit $\langle c_1, \sigma \rangle \rightarrow_1 \langle c'_1, \sigma^* \rangle$ ergibt sich dann die Behauptung für $i = i' + 1$, $j = j'$ und $\sigma' = \sigma'$. \square

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

Beweis in der Übung.

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

Beweis. Wegen $\langle c, \sigma \rangle \xrightarrow{*}_1 \langle \text{skip}, \sigma' \rangle$ gibt es ein n mit $\langle c, \sigma \rangle \xrightarrow{n}_1 \langle \text{skip}, \sigma' \rangle$. 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 $\langle c, \sigma \rangle \xrightarrow{m}_1 \langle \text{skip}, \sigma' \rangle$, dann auch $\langle c, \sigma \rangle \Downarrow \sigma'$. Zu zeigen: Aus (i) $\langle c, \sigma \rangle \xrightarrow{n}_1 \langle \text{skip}, \sigma' \rangle$ folgt $\langle c, \sigma \rangle \Downarrow \sigma'$ für beliebige c, σ und σ' .

Fallunterscheidung nach c :

- Fall $c = \text{skip}$: Mit (i) folgt, dass $n = 0$ und $\sigma' = \sigma$. $\langle \text{skip}, \sigma \rangle \Downarrow \sigma$ folgt aus Regel SKIP_{BS}.
- Fall $c = x := a$: Mit (i) folgt, dass $n = 1$ und $\sigma' = \sigma[x \mapsto \mathcal{A} \llbracket a \rrbracket \sigma]$. $\langle x := a, \sigma \rangle \Downarrow \sigma[x \mapsto \mathcal{A} \llbracket a \rrbracket \sigma]$ folgt aus der Regel ASS_{BS}.
- Fall $c = c_1; c_2$:
Nach dem Zerlegungslemma lässt sich (i) in $\langle c_1, \sigma \rangle \xrightarrow{i}_1 \langle \text{skip}, \sigma^* \rangle$ und $\langle c_2, \sigma^* \rangle \xrightarrow{j}_1 \langle \text{skip}, \sigma' \rangle$ 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 SEQ_{BS}.
- Fall $c = \text{if } (b) \text{ then } c_1 \text{ else } c_2$: Aus (i) folgt mit Regelinversion, dass $n > 0$ und entweder (IFTT_{SS}) $\mathcal{B} \llbracket b \rrbracket \sigma = \text{tt}$ und $\langle c_1, \sigma \rangle \xrightarrow{n-1}_1 \langle \text{skip}, \sigma' \rangle$ oder (IFFF_{SS}) $\mathcal{B} \llbracket b \rrbracket \sigma = \text{ff}$ und $\langle c_2, \sigma \rangle \xrightarrow{n-1}_1 \langle \text{skip}, \sigma' \rangle$. In beiden Fällen lässt sich die Induktionsannahme anwenden und die Behauptung folgt aus den Regeln IFTT_{BS} bzw. IFFF_{BS}.
- Fall $c = \text{while } (b) \text{ do } c$: Aus (i) folgt mit Regelinversion (WHILE_{SS}), dass $n > 0$ und

$$\langle \text{while } (b) \text{ do } c, \sigma \rangle \rightarrow_1 \underbrace{\langle \text{if } (b) \text{ then } c; \text{ while } (b) \text{ do } c \text{ else skip}, \sigma \rangle}_{=w'} \xrightarrow{n-1}_1 \langle \text{skip}, \sigma' \rangle.$$

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 $\langle \text{while } (b) \text{ do } c, \sigma \rangle \Downarrow \sigma'$. \square

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 $\langle c, \sigma \rangle \xrightarrow{*}_1 \langle \text{skip}, \sigma' \rangle$ gilt.