Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen
Sebastian Ullrich sebastian.ullrich@kit.edu
Max Wagner maxmilian.wagner@kit.edu

Semantik von Programmiersprachen – SS 2019

http://pp.ipd.kit.edu/lehre/SS2019/semantik

Lösungen zu Blatt 3: Small-Step-Semantik Besprechung: 13.05.2019

1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

(a) b_1 `&&` b_2 verhält sich semantisch wie `if (`b_1`) then` b_2 `else false`.

(b) $\langle$`if (`b`) then` c_1 `else` c_2, $\sigma\rangle$ und $\langle$`if (not` b`) then` c_2 `else` c_1, $\sigma\rangle$ haben die gleichen Ableitungsfolgen.

(c) Wenn $\langle c_1\texttt{;}\ c_2,\ \sigma\rangle \xrightarrow{*}_1 \langle c_1'\texttt{;}\ c_2,\ \sigma'\rangle$, dann auch $\langle c_1,\ \sigma\rangle \xrightarrow{*}_1 \langle c_1',\ \sigma'\rangle$.

(d) Wenn $\langle c,\ \sigma\rangle \xrightarrow{n}_1 \langle c,\ \sigma\rangle$, dann $n = 0$.

(e) $\langle$`x := 1; while (x <= 2) do x := x * 2`, $\sigma\rangle \xrightarrow{12}_1 \langle$`skip`, $\sigma[\texttt{x} \mapsto 4]\rangle$.

(f) Wenn $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle$`skip`, $\sigma'\rangle$, dann enthält c syntaktisch keine Schleife der Form `while (true) do` c'.

(g) Wenn $(\gamma_i)_i$ und $(\delta_j)_j$ Ableitungsfolgen für $\langle c,\ \sigma\rangle$ sind, dann $(\gamma_i)_i = (\delta_j)_j$.

(h) Wenn $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle c',\ \sigma'\rangle$, dann enthält c' höchstens dreimal so viele AST-Knoten wie c.

Lösung:

(1a) Falsch. Das würde funktionieren, wenn man ein „funktionales“ `if` hätte, allerdings ist `if (`b_1`) then` b_2 `else false` gar kein While-Programm.

(1b) Falsch. Nach dem ersten Reduktionsschritt sind die *maximalen* Ableitungsfolgen alle gleich. Das erste Element enthält jedoch auch die Anweisung, die bei beiden unterschiedlich ist.

(1c) Falsch. Die Crux hier ist, dass die WhileSS-Regel über Schleifenabwicklung funktioniert: Dass nach n Schritten $c_1'\texttt{;}\ c_2$ dasteht heißt nicht, dass c_2 in der bisherigen Ableitungsfolge uninvolviert war! Beispiel: c_1 = `x := 0`, c_2 = `while (true) do x := x + 1` und c_1' = `x := x + 1`.

$$
\begin{aligned}
&\langle c_1\texttt{;}\ c_2,\ \sigma\rangle \\
\xrightarrow{*}_1&\langle c_2,\ \sigma[\texttt{x} \mapsto 0]\rangle \\
\to_1&\langle \texttt{if (true) then (}c_1'\texttt{;}\ c_2\texttt{) else skip},\ \sigma[\texttt{x} \mapsto 0]\rangle \\
\to_1&\langle c_1'\texttt{;}\ c_2,\ \sigma[\texttt{x} \mapsto 0]\rangle \\
\xrightarrow{*}_1&\langle c_1'\texttt{;}\ c_2,\ \sigma[\texttt{x} \mapsto 1]\rangle \\
&\dots \\
\xrightarrow{*}_1&\langle c_1'\texttt{;}\ c_2,\ \sigma[\texttt{x} \mapsto 42]\rangle
\end{aligned}
$$

Jedoch gilt für kein σ', dass $\langle c_1,\ \sigma\rangle \xrightarrow{*}_1 \langle c_1',\ \sigma'\rangle$.

(1d) Falsch. Gegenbeispiel: $c = \texttt{while (true) do skip}$, σ beliebig.

$$\begin{aligned}&\langle c,\ \sigma\rangle \to_1 \langle \texttt{if (true) then skip; while (true) do skip else skip},\ \sigma\rangle\\ \to_1 &\langle \texttt{skip; while (true) do skip},\ \sigma\rangle \to_1 \langle c,\ \sigma\rangle\end{aligned}$$

Somit: $\langle c,\ \sigma\rangle \overset{3}{\to}_1 \langle c,\ \sigma\rangle$.

(1e) Richtig. Sei $c = \texttt{x := x * 2}$, $w = \texttt{while (x <= 2) do } c$,
$w' = \texttt{if (x <= 2) then } c\texttt{; } w \texttt{ else skip}$ und $\sigma_n = \sigma[\texttt{x} \mapsto n]$.

$$\begin{aligned}&\langle \texttt{x := 1; } w,\ \sigma\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_1\rangle\\ \to_1 &\langle w,\ \sigma_1\rangle \to_1 \langle w',\ \sigma_1\rangle \to_1 \langle c\texttt{; } w,\ \sigma_1\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_2\rangle\\ \to_1 &\langle w,\ \sigma_2\rangle \to_1 \langle w',\ \sigma_2\rangle \to_1 \langle c\texttt{; } w,\ \sigma_2\rangle \to_1 \langle \texttt{skip; } w,\ \sigma_4\rangle\\ \to_1 &\langle w,\ \sigma_4\rangle \to_1 \langle w',\ \sigma_4\rangle \to_1 \langle \texttt{skip},\ \sigma_4\rangle\end{aligned}$$

(1f) Falsch. Gegenbeispiel:

$$\langle \texttt{if (false) then while (true) do skip else skip},\ \sigma\rangle \to_1 \langle \texttt{skip},\ \sigma\rangle$$

(1g) Falsch, dies gilt nur für maximale Ableitungsfolgen. Bei normalen Ableitungsfolgen kann die eine ein Präfix der anderen sein.

(1h) Falsch. Für jedes Programm gibt es zwar eine maximal erreichbare Größe, das Verhältnis ist aber nicht beschränkt. Gegenbeispiel für 3:

```
while (true) do while (true) do while (true) do (skip)
```

2. Small-Step simuliert Big-Step (H)

Beweisen Sie durch Induktion über die Regeln der Big-Step-Semantik, dass jede Ausführung in der Big-Step-Semantik eine äquivalente Ausführung in der Small-Step-Semantik besitzt, d.h.: Aus $\langle c,\ \sigma\rangle \Downarrow \sigma'$ folgt $\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$.

Hinweis: Für die $\textsc{Seq}_{\text{BS}}$- und $\textsc{WhileTT}_{\text{BS}}$-Fälle benötigen Sie Lemma 19 aus dem Skript.

Lösung:

Beweis. Beweis per Induktion über $\langle c,\ \sigma\rangle \Downarrow \sigma'$:

- Fall $\textsc{Skip}_{\text{BS}}$: Wir haben $c = \texttt{skip}$ und $\sigma' = \sigma$.
  Zu zeigen: $\langle \texttt{skip},\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma\rangle$. Trivial ($* = 0$).
- Fall $\textsc{Ass}_{\text{BS}}$: Wir haben $c = x\ \texttt{:=}\ a$ und $\sigma' = \sigma[\texttt{x} \mapsto \mathcal{A}[\![a]\!]\sigma]$.
  Zu zeigen: $\langle \texttt{x := } a,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma[\texttt{x} \mapsto \mathcal{A}[\![a]\!]\sigma]\rangle$. Dies folgt aus Regel $\textsc{Ass}_{\text{SS}}$ mit $* = 1$.
- Fall $\textsc{Seq}_{\text{BS}}$: Induktionsannahmen: (i) $\langle c_1,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ und (ii) $\langle c_2,\ \sigma'\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$.
  Zu zeigen: $\langle c_1\texttt{; } c_2,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma''\rangle$.
  Aus (i) folgt nach dem Liftinglemma 19 für Sequenz, dass auch $\langle c_1\texttt{; } c_2,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip; } c_2,\ \sigma'\rangle$ gilt. Weiterhin gilt $\langle \texttt{skip; } c_2,\ \sigma'\rangle \to_1 \langle c_2,\ \sigma'\rangle$ nach Regel $\textsc{Seq2}_{\text{SS}}$. Zusammen mit (ii) folgt die Behauptung aus der Transitivität von $\overset{*}{\to}_1$.
- Fall $\textsc{IfTT}_{\text{BS}}$: Induktionsannahmen: (i) $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$ und (ii) $\langle c_1,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$.
  Zu zeigen: $\langle \texttt{if } (b) \texttt{ then } c_1 \texttt{ else } c_2,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$.
  Wegen (i) gilt $\langle \texttt{if } (b) \texttt{ then } c_1 \texttt{ else } c_2,\ \sigma\rangle \to_1 \langle c_1,\ \sigma\rangle$ nach Regel $\textsc{IfTT}_{\text{SS}}$. Mit (ii) folgt die Behauptung.
- Fall $\textsc{IfFF}_{\text{BS}}$: Analog.
- Fall $\textsc{WhileFF}_{\text{BS}}$: Induktionsannahme: $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$.
  Zu zeigen: $\langle \texttt{while } (b) \texttt{ do } c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma\rangle$.
  Es gilt: $\langle \texttt{while } (b) \texttt{ do } c,\ \sigma\rangle \to_1 \langle \texttt{if } (b) \texttt{ then } c\texttt{; while } (b) \texttt{ do } c \texttt{ else skip},\ \sigma\rangle$ nach Regel $\textsc{While}_{\text{SS}}$ und $\cdots \to_1 \langle \texttt{skip},\ \sigma\rangle$ nach Regel $\textsc{IfFF}_{\text{SS}}$ mit der Induktionsannahme. Damit folgt die Behauptung.

- Fall WhileTT$_{\text{BS}}$: Induktionsannahmen: (i) $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$, (ii) $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma'\rangle$ und (iii) $\langle \texttt{while (}b\texttt{) do }c,\ \sigma'\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma''\rangle$. Zu zeigen: $\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma''\rangle$. Beweis mit dem Liftinglemma für Sequenz (Lem. 19):

$$\begin{aligned}&\langle \texttt{while (}b\texttt{) do }c,\ \sigma\rangle \to_1 \langle \texttt{if (}b\texttt{) then }c\texttt{; while (}b\texttt{) do }c\texttt{ else skip},\ \sigma\rangle\\ &\to_1 \langle c\texttt{; while (}b\texttt{) do }c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip; while (}b\texttt{) do }c,\ \sigma'\rangle \to_1 \langle \texttt{while (}b\texttt{) do }c,\ \sigma'\rangle\\ &\xrightarrow{*}_1 \langle \texttt{skip},\ \sigma''\rangle\end{aligned}$$

□

3. Operationale Small-Step-Semantik für Ausdrücke (Ü)

Die Big-Step und Small-Step-Semantiken verwenden für Ausdrücke die Semantikfunktionen $\mathcal{A}[\![\cdot]\!]$ und $\mathcal{B}[\![\cdot]\!]$, die für alle Ausdrücke und Zustände definiert sind und in einem Schritt das Ergebnis liefern. In dieser Aufgabe sollen auch diese Ausdrücke mit einer Small-Step-Semantik schrittweise ausgewertet werden.

(a) Definieren Sie eine Einschrittauswertungsrelation $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$ für arithmetische Ausdrücke und entsprechend $\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle$ für boolesche Ausdrücke, die einen einzelnen Schritt der Auswertung des Ausdrucks a bzw. b im Zustand σ zum Ausdruck a' bzw. b' beschreibt.

(b) Passen Sie die Regeln der Small-Step-Semantik $\langle\cdot,\ \cdot\rangle \to_1 \langle\cdot,\ \cdot\rangle$ an, sodass diese die neuen Relationen $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$ und $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{B}} \langle\cdot\rangle$ an Stelle von $\mathcal{A}[\![\cdot]\!]$ und $\mathcal{B}[\![\cdot]\!]$ verwendet.

(c) Identifizieren Sie die blockierten Ausdrücke bezüglich $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$ und $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{B}} \langle\cdot\rangle$. Zeigen Sie mit einem Fortschrittslemma, dass diese die einzigen blockierten Ausdrücke sind.

(d) Sind Ihre Small-Step-Semantiken deterministisch?

(e) In welcher Beziehung stehen Ihre Small-Step-Semantiken zu den Auswertungsfunktionen $\mathcal{A}[\![\cdot]\!]$ und $\mathcal{B}[\![\cdot]\!]$? Drücken Sie diese Beziehung formal aus. Überlegen Sie sich, wie Sie diese Beziehung beweisen könnten.

Lösung:

(3a) Gestaltungsspielraum hat man bei den Semantik-Regeln über die Auswertungsreihenfolge der Ausdrücke. Dies wird sich besonders bei Aufgabe (3d) zeigen. Hier die Regeln, die zuerst den linken und dann den rechten Teilausdruck auswerten:

$$\textsc{Var}\colon \sigma \vdash \langle x\rangle \to_{\mathsf{A}} \langle \mathcal{N}^{-1}[\![\sigma(x)]\!]\rangle$$

$$\textsc{Minus1}\colon \frac{\sigma \vdash \langle a_1\rangle \to_{\mathsf{A}} \langle a_1'\rangle}{\sigma \vdash \langle a_1\ \texttt{-}\ a_2\rangle \to_{\mathsf{A}} \langle a_1'\ \texttt{-}\ a_2\rangle} \qquad \textsc{Minus2}\colon \frac{\sigma \vdash \langle a_2\rangle \to_{\mathsf{A}} \langle a_2'\rangle}{\sigma \vdash \langle n_1\ \texttt{-}\ a_2\rangle \to_{\mathsf{A}} \langle n_1\ \texttt{-}\ a_2'\rangle}$$

$$\textsc{Minus}\colon \sigma \vdash \langle n_1\ \texttt{-}\ n_2\rangle \to_{\mathsf{A}} \langle \mathcal{N}^{-1}[\![\mathcal{N}[\![n_1]\!] - \mathcal{N}[\![n_2]\!]]\!]\rangle$$

$$\textsc{Times1}\colon \frac{\sigma \vdash \langle a_1\rangle \to_{\mathsf{A}} \langle a_1'\rangle}{\sigma \vdash \langle a_1\ \texttt{*}\ a_2\rangle \to_{\mathsf{A}} \langle a_1'\ \texttt{*}\ a_2\rangle} \qquad \textsc{Times2}\colon \frac{\sigma \vdash \langle a_2\rangle \to_{\mathsf{A}} \langle a_2'\rangle}{\sigma \vdash \langle n_1\ \texttt{*}\ a_2\rangle \to_{\mathsf{A}} \langle n_1\ \texttt{*}\ a_2'\rangle}$$

$$\textsc{Times}\colon \sigma \vdash \langle n_1\ \texttt{*}\ n_2\rangle \to_{\mathsf{A}} \langle \mathcal{N}^{-1}[\![\mathcal{N}[\![n_1]\!] \cdot \mathcal{N}[\![n_2]\!]]\!]\rangle$$

$$\textsc{Leq1}\colon \frac{\sigma \vdash \langle a_1\rangle \to_{\mathsf{A}} \langle a_1'\rangle}{\sigma \vdash \langle a_1\ \texttt{<=}\ a_2\rangle \to_{\mathsf{B}} \langle a_1'\ \texttt{<=}\ a_2\rangle} \qquad \textsc{Leq2}\colon \frac{\sigma \vdash \langle a_2\rangle \to_{\mathsf{A}} \langle a_2'\rangle}{\sigma \vdash \langle n_1\ \texttt{<=}\ a_2\rangle \to_{\mathsf{B}} \langle n_1\ \texttt{<=}\ a_2'\rangle}$$

$$\textsc{LeqTT}\colon \frac{\mathcal{N}[\![n_1]\!] \le \mathcal{N}[\![n_2]\!]}{\sigma \vdash \langle n_1\ \texttt{<=}\ n_2\rangle \to_{\mathsf{B}} \langle \texttt{true}\rangle} \qquad \textsc{LeqFF}\colon \frac{\mathcal{N}[\![n_1]\!] > \mathcal{N}[\![n_2]\!]}{\sigma \vdash \langle n_1\ \texttt{<=}\ n_2\rangle \to_{\mathsf{B}} \langle \texttt{false}\rangle}$$

$$\textsc{Not}: \frac{\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle}{\sigma \vdash \langle \texttt{not } b\rangle \to_{\mathsf{B}} \langle \texttt{not } b'\rangle}$$

$$\textsc{NotFF}: \sigma \vdash \langle \texttt{not false}\rangle \to_{\mathsf{B}} \langle \texttt{true}\rangle$$

$$\textsc{And}: \frac{\sigma \vdash \langle b_1\rangle \to_{\mathsf{B}} \langle b_1'\rangle}{\sigma \vdash \langle b_1 \texttt{ \&\& } b_2\rangle \to_{\mathsf{B}} \langle b_1' \texttt{ \&\& } b_2\rangle}$$

$$\textsc{AndTT}: \sigma \vdash \langle \texttt{true \&\& } b_2\rangle \to_{\mathsf{B}} \langle b_2\rangle \qquad \textsc{AndFF}: \sigma \vdash \langle \texttt{false \&\& } b_2\rangle \to_{\mathsf{B}} \langle \texttt{false}\rangle$$

Literale (n, `true`, `false`) können nicht auswerten, diese übernehmen die Rolle des `skip` in der Small-Step-Semantik für Anweisungen. Man beachte, dass nach Vorlesung `false` syntaktischer Zucker für `not true` ist, weswegen wir keine Ableitung für letzteren Term brauchen – mit der Ableitung $\sigma \vdash \langle \texttt{not true}\rangle \to_{\mathsf{B}} \langle \texttt{false}\rangle$ würden wir uns nur Nichttermination einhandeln.

Die Auswertungsreihenfolge der Operanden von a_1 `-` a_2, a_1 `*` a_2 und a_1 `<=` a_2 ist dadurch festgelegt, dass Minus2, Times2 und Leq2 in der Konklusion im ersten Operanden ein Zahlliteral n_1 erwarten. Will man eine andere Auswertungsreihenfolge modellieren, muss man Literale an entsprechend anderen Stellen einsetzen. Möchte man die Reihenfolge der Auswertung der Operanden frei lassen und diese auch verschänkt auswerten können, dann ersetzt man in diesen Regeln das n_1 durch das allgemeinere a_1. Dann muss man nur beim Determinismus aufpassen.

Hätte man eine syntaktische Darstellung für Zahlen (z.B. Binärstrings), könnte man entsprechend Subtraktion und Multiplikation von Zahlliteralen implementieren und müsste nicht den Umweg über den semantischen Bereich mittels $\mathcal{N}[\![\cdot]\!]$ und $\mathcal{N}^{-1}[\![\cdot]\!]$ nehmen. Da es nur die beiden booleschen Literale `true` und `false` gibt, ist dies für Bexp in den obigen Regeln schon geschehen.

(3b) Es müssen nur die Regeln $\textsc{Ass}_{\text{SS}}$, $\textsc{IfTT}_{\text{SS}}$ und $\textsc{IfFF}_{\text{SS}}$ durch folgende ersetzt werden:

$$\textsc{Ass1}_{\text{SS}}: \frac{\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle}{\langle x \texttt{ := } a,\ \sigma\rangle \to_1 \langle x \texttt{ := } a',\ \sigma\rangle} \qquad \textsc{Ass2}_{\text{SS}}: \langle x \texttt{ := } n,\ \sigma\rangle \to_1 \langle \texttt{skip},\ \sigma[x \mapsto \mathcal{N}[\![n]\!]]\rangle$$

$$\textsc{If}_{\text{SS}}: \frac{\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle}{\langle \texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2,\ \sigma\rangle \to_1 \langle \texttt{if (}b'\texttt{) then } c_1 \texttt{ else } c_2,\ \sigma\rangle}$$

$$\textsc{IfTT}_{\text{SS}}: \langle \texttt{if (true) then } c_1 \texttt{ else } c_2,\ \sigma\rangle \to_1 \langle c_1,\ \sigma\rangle$$

$$\textsc{IfFF}_{\text{SS}}: \langle \texttt{if (false) then } c_1 \texttt{ else } c_2,\ \sigma\rangle \to_1 \langle c_2,\ \sigma\rangle$$

(3c) Die blockierten Ausdrücke sind nur die Literale n, `true` und `false`.

Fortschrittslemma für $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$: Wenn a nicht von der Form n ist, dann gibt es ein a' mit $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$.

Beweis. Induktion über a.

- Fall $a = x$: Wähle $a' = \mathcal{N}^{-1}[\![\sigma(x)]\!]$. Damit nach Regel Var.
- Fall $a = a_1$ `-` a_2: Fallunterscheidung nach a_1
  - a_1 ist nicht von der Form n_1:
    Dann gibt es nach Induktionsannahme a_1' mit $\sigma \vdash \langle a_1\rangle \to_{\mathsf{A}} \langle a_1'\rangle$. Damit folgt die Behauptung mit $a' = a_1'$ `-` a_2 und Minus1.
  - a_1 ist von der Form n_1, a_2 ist nicht von der Form n_2:
    Dann gibt es nach Induktionsannahme a_2' mit $\sigma \vdash \langle a_2\rangle \to_{\mathsf{A}} \langle a_2'\rangle$. Damit folgt die Behauptung mit $a' = n_1$ `-` a_2' und Minus2.
  - a_1 und a_2 sind von der Form n_1 und n_2:
    Dann folgt die Behauptung mit $a' = \mathcal{N}^{-1}[\![\mathcal{N}[\![n_1]\!] - \mathcal{N}[\![n_2]\!]]\!]$ und Minus.

- Fall $a = a_1$ `*` a_2: Analog zu $a = a_1$ `-` a_2. □

Fortschrittslemma für $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{B}} \langle\cdot\rangle$: Wenn $b \neq \texttt{true}$ und $b \neq \texttt{false}$, dann gibt es ein b' mit $\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle$.

Beweis. Induktion über b analog zum Fortschrittslemma für $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$. Im Fall $b = a_1$ `<=` a_2 braucht man ebendieses Fortschrittslemma, wenn a_1 oder a_2 nicht von der Form n sind. □

Beide Fortschrittslemmata bräuchte man, um das Fortschrittslemma (Lem. 16) für die geänderte Small-Step-Semantik zu beweisen.

(3d) Für die hier angegebenen Semantiken lässt sich der Einschrittdeterminismus sehr einfach zeigen:

Determinismus für $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$ Wenn $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$ und $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a''\rangle$, dann $a' = a''$.

Beweis mittels Induktion über $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$ (a'' beliebig) und Regelinversion auf dem anderen Schritt – analog zur Big-Step-Semantik.

Determinismus für $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{B}} \langle\cdot\rangle$ Wenn $\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle$ und $\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b''\rangle$, dann $b' = b''$.

Beweis wieder mit Regelinduktion und Regelinversion. Man braucht den Determinismus von $\cdot \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$ in den Fällen LEQ1 und LEQ2.

Aus dem Einschrittdeterminismus bekommt man sehr einfach wieder über Induktion, dass auch die erreichbaren blockierten Ausdrücke eindeutig sind. Mit diesen beiden Einzelschrittdeterminismusaussagen lässt sich dann auch wieder der Einschrittdeterminismus der Small-Step-Semantik zeigen.

Wenn man die Auswertungsreihenfolge nicht vorgibt, also n_1 in MINUS2 und TIMES2 durch a_1 ersetzt, dann kann man den Einschrittdeterminismus nicht mehr zeigen: Sei $\sigma = [\mathtt{x} \mapsto 0, \mathtt{y} \mapsto 1]$. Dann $\sigma \vdash \langle \texttt{x - y}\rangle \to_{\mathsf{A}} \langle \texttt{0 - y}\rangle$ und $\sigma \vdash \langle \texttt{x - y}\rangle \to_{\mathsf{A}} \langle \texttt{x - 1}\rangle$, aber $\texttt{0 - y} \neq \texttt{x - 1}$.

Trotzdem könnte man in diesem Fall die Eindeutigkeit der erreichbaren blockierten Ausdrücke (also der Literale) beweisen: Dazu zeigt man, dass $\sigma \vdash \langle\cdot\rangle \to_{\mathsf{A}} \langle\cdot\rangle$ lokal konfluent und terminierend ist, und ebenso für $\sigma \vdash \langle\cdot\rangle \to_{\mathsf{B}} \langle\cdot\rangle$. Daraus folgt dann globale Konfluenz und Eindeutigkeit der blockierten Konfigurationen. Damit könnte man dann wieder die Konfluenz der Small-Step-Semantik zeigen. Alternativ erhält man die Eindeutigkeit der erreichbaren blockierten Zustände aus (3e).

(3e) Das erreichbare Literal eines Ausdrucks entspricht dem Wert der Auswertungsfunktion:

$$\begin{array}{lll} \sigma \vdash \langle a\rangle \xrightarrow{*}_{\mathsf{A}} \langle n\rangle & \text{gdw.} & \mathcal{A}[\![a]\!]\,\sigma = \mathcal{N}[\![n]\!] \\ \sigma \vdash \langle b\rangle \xrightarrow{*}_{\mathsf{B}} \langle \texttt{true}\rangle & \text{gdw.} & \mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \\ \sigma \vdash \langle b\rangle \xrightarrow{*}_{\mathsf{B}} \langle \texttt{false}\rangle & \text{gdw.} & \mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \end{array}$$

Dieser Beweis entspricht dem Adäquatheitstheorem aus der denotationalen Semantik – letztendlich sind $\mathcal{A}[\![\cdot]\!]$ und $\mathcal{B}[\![\cdot]\!]$ denotationale Semantiken für Ausdrücke. Er nimmt an, dass jede ganze Zahl eine eindeutige syntaktische Darstellung hat, also dass $\mathcal{N}[\![\cdot]\!]$ injektiv ist.

Zu zeigende Hilfsaussagen:

i. Einschrittreduktionen ändern den Wert der Auswertungsfunktion nicht:
   - Wenn $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$, dann $\mathcal{A}[\![a]\!]\,\sigma = \mathcal{A}[\![a']\!]\,\sigma$. Beweis durch Regelinduktion über $\sigma \vdash \langle a\rangle \to_{\mathsf{A}} \langle a'\rangle$.
   - Wenn $\sigma \vdash \langle b\rangle \to_{\mathsf{B}} \langle b'\rangle$, dann $\mathcal{B}[\![b]\!]\,\sigma = \mathcal{B}[\![b']\!]\,\sigma$. Beweis analog.

ii. Verallgemeinerung dieser Aussagen auf reflexiv-transitive Ableitungen durch Induktion über die reflexiv-transitive Hülle.
   - Wenn $\sigma \vdash \langle a\rangle \xrightarrow{*}_{\mathsf{A}} \langle a'\rangle$, dann $\mathcal{A}[\![a]\!]\,\sigma = \mathcal{A}[\![a']\!]\,\sigma$.
   - Wenn $\sigma \vdash \langle b\rangle \xrightarrow{*}_{\mathsf{B}} \langle b'\rangle$, dann $\mathcal{B}[\![b]\!]\,\sigma = \mathcal{B}[\![b']\!]\,\sigma$.

   Daraus folgt direkt die Richtung von links nach rechts für alle drei Aussagen.

iii. Für die umgekehrte Richtung braucht man viele „Lifting-Lemmata“ analog zum Lifting-Lemma für Sequenz (Lem. 19) – eines für jede rekursive Regel in den Small-Step-Semantiken:

- Wenn $\sigma \vdash \langle a \rangle \xrightarrow{*}_{\mathsf{A}} \langle a' \rangle$, dann auch $\sigma \vdash \langle a \texttt{ - } a_2 \rangle \xrightarrow{*}_{\mathsf{A}} \langle a' \texttt{ - } a_2 \rangle$, $\sigma \vdash \langle n_1 \texttt{ - } a \rangle \xrightarrow{*}_{\mathsf{A}} \langle n_1 \texttt{ - } a' \rangle$, $\sigma \vdash \langle a \texttt{ * } a_2 \rangle \xrightarrow{*}_{\mathsf{A}} \langle a' \texttt{ * } a_2 \rangle$, $\sigma \vdash \langle n_1 \texttt{ * } a \rangle \xrightarrow{*}_{\mathsf{A}} \langle n_1 \texttt{ * } a' \rangle$, $\sigma \vdash \langle a \texttt{ <= } a_2 \rangle \xrightarrow{*}_{\mathsf{B}} \langle a' \texttt{ <= } a_2 \rangle$ und $\sigma \vdash \langle n_1 \texttt{ <= } a \rangle \xrightarrow{*}_{\mathsf{B}} \langle n_1 \texttt{ <= } a' \rangle$.
- Wenn $\sigma \vdash \langle b \rangle \xrightarrow{*}_{\mathsf{B}} \langle b' \rangle$, dann auch $\sigma \vdash \langle \texttt{not } b \rangle \xrightarrow{*}_{\mathsf{B}} \langle \texttt{not } b' \rangle$ und $\sigma \vdash \langle b \texttt{ \&\& } b_2 \rangle \xrightarrow{*}_{\mathsf{B}} \langle b' \texttt{ \&\& } b_2 \rangle$.

Damit kann man nun direkt die umgekehrte Richtung mittels Induktion über a bzw. b beweisen. Dies setzt im Fall $a = n$ voraus, dass $\mathcal{N}[\![\cdot]\!]$ injektiv ist.

iii'. Statt dieser Hilfslemmata kann man die Rückrichtung auch über Terminierung beweisen: Angenommen, $\mathcal{A}[\![a]\!]\sigma = \mathcal{N}[\![n]\!]$. Da $\sigma \vdash \langle \cdot \rangle \rightarrow_{\mathsf{A}} \langle \cdot \rangle$ terminierend ist, gibt es ein n' mit $\sigma \vdash \langle a \rangle \xrightarrow{*}_{\mathsf{A}} \langle n' \rangle$. Damit gilt nach dem soeben bewiesenen, dass $\mathcal{A}[\![a]\!]\sigma = \mathcal{N}[\![n']\!] = \mathcal{N}[\![n]\!]$. Da $\mathcal{N}[\![\cdot]\!]$ injektiv ist, gilt also $n = n'$.

Wenn $\mathcal{N}[\![\cdot]\!]$ nicht injektiv ist (weil es mehrere syntaktische Darstellungen einer Zahl gibt), dann folgt aus $\mathcal{A}[\![a]\!]\sigma = n$ nur, dass es ein n' mit $\sigma \vdash \langle a \rangle \rightarrow_{\mathsf{A}} \langle n' \rangle$ und $\mathcal{N}[\![n']\!] = n$ gibt