Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen

Andreas Lochbihler
andreas.lochbihler@kit.edu

Joachim Breitner
breitner@kit.edu

Semantik von Programmiersprachen – SS 2012

http://pp.info.uni-karlsruhe.de/lehre/SS2012/semantik

Lösungen zu Blatt 4: Big-Step- und Small-Step-Semantik Besprechung: 15.05.2012

1. Semantik mit Ausführungszeiten (H)

Die Big-Step-Semantik für While berücksichtigt nicht, wie viele Schritte die Ausführung eines Programms benötigt. Dies sollen Sie in dieser Aufgabe modellieren:

(a) Definieren Sie, auf Papier oder in Prolog, eine Auswertungsrelation $\langle c,\ \sigma\rangle \Downarrow_t \sigma'$ mit der Bedeutung, dass die Ausführung von c im Anfangszustand σ im Endzustand σ' endet und dafür t Schritte benötigt. Finden Sie dafür eine geeignete Definition, was ein Schritt sein soll.

(b) Beschreiben Sie formal, in welcher Beziehung $\langle c,\ \sigma\rangle \Downarrow \sigma'$ und $\langle c,\ \sigma\rangle \Downarrow_t \sigma'$ stehen. Wie würde man diese Beziehung beweisen?

(c) Überprüfen Sie, welche der folgenden Eigenschaften der Big-Step-Semantik $\langle c,\ \sigma\rangle \Downarrow \sigma'$ sich auf $\langle c,\ \sigma\rangle \Downarrow_t \sigma'$ übertragen lassen. Formulieren Sie die Aussagen entsprechend. Wie müssten die Beweise angepasst werden?

- Schleifenabwicklungslemma (Lem. 1)
- Determinismus (Thm. 2)
- Äquivalenz zwischen Big-Step- und Small-Step-Semantik (Kor. 10)

Lösung:

(1a) Verschiedene Begriffe für einen Schritt sind denkbar:

- Jeder Ableitungsschritt im Baum ist ein Schritt. Dies ist der natürlichste Ansatz bezüglich der Big-Step-Semantik, erlaubt aber nur schwierige Vergleiche mit den Längen der Ableitungsfolgen der Small-Step-Semantik. Die Auswertung eines arithmetischen oder booleschen Ausdrucks kostet in diesem Modell keine Zeit.
- Man definiert auch Ausführungszeiten für arithmetische und boolesche Ausdrücke, z.B. mit einer rekursiven Ausführungszeitfunktion. Dann muss man allerdings neu über syntaktischen Zucker für arithmetische und boolesche Ausdrücke nachdenken: Eine Addition würde dann ggf. länger dauern als eine Subtraktion!
- Man definiert die Ausführungszeiten so, dass genau die Länge der Small-Step-Semantik herauskommt – oder die Zahl der Ausführungsschritte in ASM nach Übersetzung mit dem Compiler. Damit kann man dann zwar schöne Äquivalenz- bzw. Erhaltungseigenschaften zeigen, allerdings ist dieser Zeitbegriff sehr künstlich.
  Nielson und Nielson scheinen (Kap. 10.2, Tabelle 10.3) eine solche Definition bezüglich der Small-Step-Semantik anzugeben.

Hier zuerst die einfachste Version. Die Regeln für die Big-Step-Semantik ergeben sich quasi direkt aus den originalen Regeln:

$$\textsc{Skip}_{\text{TBS}}\colon \langle \texttt{skip},\ \sigma\rangle \Downarrow_1 \sigma \qquad \textsc{Ass}_{\text{TBS}}\colon \langle x\ \texttt{:=}\ a,\ \sigma\rangle \Downarrow_1 \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]$$

$$\textsc{Seq}_{\text{TBS}}\colon \frac{\langle c_0,\ \sigma\rangle \Downarrow_t \sigma' \qquad \langle c_1,\ \sigma'\rangle \Downarrow_{t'} \sigma''}{\langle c_0\texttt{;}\ c_1,\ \sigma\rangle \Downarrow_{t+t'+1} \sigma''}$$

$$\textsc{IfTT}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c_0,\ \sigma\rangle \Downarrow_t \sigma'}{\langle \texttt{if (}b\texttt{) then}\ c_0\ \texttt{else}\ c_1,\ \sigma\rangle \Downarrow_{t+1} \sigma'}$$

$$\textsc{IfFF}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \qquad \langle c_1,\ \sigma\rangle \Downarrow_t \sigma'}{\langle \texttt{if (}b\texttt{) then}\ c_0\ \texttt{else}\ c_1,\ \sigma\rangle \Downarrow_{t+1} \sigma'}$$

$$\textsc{WhileFF}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}}{\langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_1 \sigma}$$

$$\textsc{WhileTT}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c,\ \sigma\rangle \Downarrow_t \sigma' \qquad \langle \texttt{while (}b\texttt{) do}\ c,\ \sigma'\rangle \Downarrow_{t'} \sigma''}{\langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_{t+t'+1} \sigma''}$$

Oder als Prolog-Programm:

```
evalTBS(skp, S, S, 1).
evalTBS((P1;P2), S1, S3, T) :-
        evalTBS(P1, S1, S2, T1),
        evalTBS(P2, S2, S3, T2),
        T is T1 + T2 + 1.
evalTBS(Var := Aexp, S1, S2, 1) :-
        evalA(S1, Aexp, Value),
        set(S1, Var, Value, S2).
evalTBS(cond(Bexp, P1, _), S1, S2, T) :-
        evalB(S1, Bexp, tt),
        evalTBS(P1, S1, S2, T1),
        T is T1 + 1.
evalTBS(cond(Bexp, _, P2), S1, S2, T) :-
        evalB(S1, Bexp, ff),
        evalTBS(P2, S1, S2, T2),
        T is T2 + 1.
evalTBS(while(Bexp,_), S1, S1, 1) :-
        evalB(S1, Bexp, ff).
evalTBS(while(Bexp,P), S1, S3, T) :-
        evalB(S1, Bexp, tt),
        evalTBS(P, S1, S2, T1),
        evalTBS(while(Bexp,P), S2, S3, T2),
        T is T1 + T2 + 1.
```

Wenn man auch die Auswertungszeit der Ausdrücke einbeziehen möchte, bräuchte man noch folgende Funktionen:

$$\begin{aligned}
\mathcal{TA}[\![n]\!] &= 1\\
\mathcal{TA}[\![x]\!] &= 1\\
\mathcal{TA}[\![a_1\ \texttt{-}\ a_2]\!] &= \mathcal{TA}[\![a_1]\!] + \mathcal{TA}[\![a_2]\!] + 1\\
\mathcal{TA}[\![a_1\ \texttt{*}\ a_2]\!] &= \mathcal{TA}[\![a_1]\!] + \mathcal{TA}[\![a_2]\!] + 1
\end{aligned}$$

$$\begin{aligned}
\mathcal{TB}[\![\texttt{true}]\!] &= 1\\
\mathcal{TB}[\![\texttt{false}]\!] &= 1\\
\mathcal{TB}[\![a_1\ \texttt{<=}\ a_2]\!] &= \mathcal{TA}[\![a_1]\!] + \mathcal{TA}[\![a_2]\!] + 1\\
\mathcal{TB}[\![\texttt{not}\ b]\!] &= \mathcal{TB}[\![b]\!] + 1\\
\mathcal{TB}[\![b_1\ \texttt{\&\&}\ b_2]\!] &= \mathcal{TB}[\![b_1]\!] + \mathcal{TB}[\![b_2]\!] + 1
\end{aligned}$$

Damit sähen die Big-Step-Regeln dann so aus:

$$\textsc{Skip}_{\text{TBS}}\colon \langle \texttt{skip},\ \sigma\rangle \Downarrow_1 \sigma \qquad \textsc{Ass}_{\text{TBS}}\colon \langle x\ \texttt{:=}\ a,\ \sigma\rangle \Downarrow_{\mathcal{TA}[\![a]\!]+1} \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]$$

$$\textsc{Seq}_{\text{TBS}}\colon \frac{\langle c_0,\ \sigma\rangle \Downarrow_t \sigma' \qquad \langle c_1,\ \sigma'\rangle \Downarrow_{t'} \sigma''}{\langle c_0\texttt{;}\ c_1,\ \sigma\rangle \Downarrow_{t+t'+1} \sigma''}$$

$$\textsc{IfTT}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c_0,\ \sigma\rangle \Downarrow_t \sigma'}{\langle \texttt{if (}b\texttt{) then}\ c_0\ \texttt{else}\ c_1,\ \sigma\rangle \Downarrow_{\mathcal{TB}[\![b]\!]+t+1} \sigma'}$$

$$\textsc{IfFF}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \qquad \langle c_1,\ \sigma\rangle \Downarrow_t \sigma'}{\langle \texttt{if (}b\texttt{) then}\ c_0\ \texttt{else}\ c_1,\ \sigma\rangle \Downarrow_{\mathcal{TB}[\![b]\!]+t+1} \sigma'}$$

$$\textsc{WhileFF}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}}{\langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_{\mathcal{TB}[\![b]\!]+1} \sigma}$$

$$\textsc{WhileTT}_{\text{TBS}}\colon \frac{\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \qquad \langle c,\ \sigma\rangle \Downarrow_t \sigma' \qquad \langle \texttt{while (}b\texttt{) do}\ c,\ \sigma'\rangle \Downarrow_{t'} \sigma''}{\langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_{\mathcal{TB}[\![b]\!]+t+t'+1} \sigma''}$$

(1b) Es gilt: $\langle c,\ \sigma\rangle \Downarrow \sigma'$ gdw. $\exists t.\ \langle c,\ \sigma\rangle \Downarrow_t \sigma'$. Beweis der einzelnen Richtungen durch Regelinduktion.

(1c) Alle drei Eigenschaften lassen sich über die Äquivalenz aus (1b) sogar ohne zusätzlichen Beweisaufwand übertragen, dann bekommt man aber recht schwache Aussagen, die keinerlei Informationen über die Ausführungszeiten beinhalten:

Schleifenabwicklungslemma
$\exists t.\ \langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_t \sigma'$
gdw.
$\exists t.\ \langle \texttt{if (}b\texttt{) then (}c\texttt{; while (}b\texttt{) do}\ c\texttt{) else skip},\ \sigma\rangle \Downarrow_t \sigma'$.

Determinismus Wenn $\langle c,\ \sigma\rangle \Downarrow_t \sigma'$ und $\langle c,\ \sigma\rangle \Downarrow_{t'} \sigma''$, dann $\sigma' = \sigma'$.

Äquivalenz zu Small-Step $\exists t.\ \langle c,\ \sigma\rangle \Downarrow_t \sigma'$ gdw. $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma'\rangle$.

Es sind aber auch folgende stärkere Aussagen möglich – die aber von dem gewählten Schrittbegriff abhängen und hier nur für den einfachen Fall (ein Schritt = eine Ableitungsregelanwendung) angegeben sind:

Schleifenabwicklungslemma
$\langle \texttt{while (}b\texttt{) do}\ c,\ \sigma\rangle \Downarrow_t \sigma'$
gdw.
$\langle \texttt{if (}b\texttt{) then (}c\texttt{; while (}b\texttt{) do}\ c\texttt{) else skip},\ \sigma\rangle \Downarrow_{t+1} \sigma'$.
Der Beweis dazu funktioniert analog zum Schleifenabwicklungslemma (Lem. 1) mittels „Ableitungsbaumtransformation“, nur dass hierbei auch die Zeiten mitberücksichtigt werden müssen.

Determinismus Wenn $\langle c,\ \sigma\rangle \Downarrow_t \sigma'$ und $\langle c,\ \sigma\rangle \Downarrow_{t'} \sigma''$, dann $\sigma' = \sigma''$ und $t = t'$.
Auch hier lässt sich der alte Beweis (Lem. 2) direkt übertragen: Induktion über eine Ableitung zusammen mit Regelinversion auf der anderen Ableitung.

Äquivalenz zu Small-Step Selbst wenn es verlockend wäre, aus der Zahl der Ableitungsschritte der Small-Step-Semantik die Ausführungdauer in der Big-Step-Semantik exact berechnen zu wollen (oder umgekehrt), geht dies nicht, ohne die Struktur des Programms anzusehen – sofern man nicht von vornherein die Schrittanzahl in der Small-Step-Semantik zugrundegelegt hat.
Man kann allerdings Abschätzungen beweisen, die zeigen, dass die Ausführungszeiten zumindest asymptotisch gleich sind: Aus $\langle c,\ \sigma\rangle \Downarrow_{t_1} \sigma'$ folgt $\langle c,\ \sigma\rangle \overset{t_2}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ mit $t_2 \leq 3 \cdot t_1$ und aus $\langle c,\ \sigma\rangle \overset{t_2}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ folgt $\langle c,\ \sigma\rangle \Downarrow_{t_1} \sigma'$ mit $t_1 \leq 2 \cdot t_2 + 1$. Der Beweis folgt der Struktur der Äquivalenz der Semantiken ohne Ausführungszeiten.

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

Die Big-Step und Small-Step-Semantiken verwenden für Ausdrücke die Semantikfunktionen $\mathcal{A}[\![\_]\!]$ und $\mathcal{B}[\![\_]\!]$, 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 $\langle \_,\ \_\rangle \to_1 \langle \_,\ \_\rangle$ an, sodass diese die neuen Relationen $\_ \vdash \langle \_\rangle \to_{\mathsf{A}} \langle \_\rangle$ und $\_ \vdash \langle \_\rangle \to_{\mathsf{B}} \langle \_\rangle$ an Stelle von $\mathcal{A}[\![\_]\!]$ und $\mathcal{B}[\![\_]\!]$ verwendet.

(c) Identifizieren Sie die blockierten Ausdrücke bezüglich $\_ \vdash \langle \_\rangle \to_{\mathsf{A}} \langle \_\rangle$ und $\_ \vdash \langle \_\rangle \to_{\mathsf{B}} \langle \_\rangle$. 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 $\mathcal{A}[\![\_]\!]$ und $\mathcal{B}[\![\_]\!]$? Drücken Sie diese Beziehung formal aus. Überlegen Sie sich, wie Sie diese Beziehung beweisen könnten.

Lösung:

(2a) Gestaltungsspielraum hat man bei den Semantik-Regeln über die Auswertungsreihenfolge der Ausdrücke. Dies wird sich besonders bei Aufgabe (2d) zeigen. Hier die Regeln, die zuerst den linken und dann den rechten Teilausdruck auswerten:

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

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

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

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

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

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

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

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

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

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

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

Literale (n, `true`, `false`) können nicht auswerten, diese übernehmen die Rolle des `skip` in der Small-Step-Semantik für Anweisungen. 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 entsprechend zu $\mathcal{N}^{-1}[\![\_]\!]$ eine Funktion, die boolesche Wahrheitswerte aus $\mathbb{B}$ auf $\{$ `true`, `false` $\}$ abbildet, könnte man die Reduktionsregeln LeqTT und LeqFF bzw. NotTT und NotFF für n_1 `<=` n_2 bzw. `not` _ zusammenfassen – analog zu Minus und Times.

Hätte man auch 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 $\mathcal{N}[\![\_]\!]$ und $\mathcal{N}^{-1}[\![\_]\!]$ nehmen. Da es nur die beiden booleschen Literale `true` und `false` gibt, ist dies in den obigen Regeln schon geschehen.

(2b) Es müssen nur die Regeln Ass_{SS}, IfTT_{SS} und IfFF_{SS} durch folgende ersetzt werden:

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

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

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

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

(2c) Die blockierten Ausdrücke sind nur die Literale n, `true` und `false`.

Fortschrittslemma für $\_ \vdash \langle\_\rangle \to_{\mathsf{A}} \langle\_\rangle$**:** 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 $\_ \vdash \langle \_ \rangle \to_{\mathbf{B}} \langle \_ \rangle$**:** Wenn $b \neq$ `true` und $b \neq$ `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 $\_ \vdash \langle \_ \rangle \to_{\mathsf{A}} \langle \_ \rangle$. 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. 3) für die geänderte Small-Step-Semantik zu beweisen.

(2d) Für die hier angegebenen Semantiken lässt sich der Einschrittdeterminismus sehr einfach zeigen:

Determinismus für $\_ \vdash \langle \_ \rangle \to_{\mathbf{A}} \langle \_ \rangle$ 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 $\_ \vdash \langle \_ \rangle \to_{\mathbf{B}} \langle \_ \rangle$ 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 $\_ \vdash \langle \_ \rangle \to_{\mathsf{A}} \langle \_ \rangle$ 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 $\sigma \vdash \langle \mathtt{x - y} \rangle \to_{\mathsf{A}} \langle \mathtt{0 - y} \rangle$ und $\sigma \vdash \langle \mathtt{x - y} \rangle \to_{\mathsf{A}} \langle \mathtt{x - 1} \rangle$, aber $\mathtt{0 - y} \neq \mathtt{x - 1}$.

Trotzdem könnte man in diesem Fall die Eindeutigkeit der erreichbaren blockierten Ausdrücke (also der Literale) beweisen: Dazu zeigt man, dass $\sigma \vdash \langle \_ \rangle \to_{\mathsf{A}} \langle \_ \rangle$ lokal konfluent und terminierend ist, und ebenso für $\sigma \vdash \langle \_ \rangle \to_{\mathsf{B}} \langle \_ \rangle$. 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 (2e).

(2e) Das erreichbare Literal eines Ausdrucks entspricht dem Wert der Auswertungsfunktion:

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

Dieser Beweis entspricht dem Adäquatheitstheorem aus der denotationalen Semantik – letztendlich sind $\mathcal{A}[\![\_]\!]$ und $\mathcal{B}[\![\_]\!]$ denotationale Semantiken für Ausdrücke. Er nimmt an, dass jede ganze Zahl eine eindeutige syntaktische Darstellung hat, also dass $\mathcal{N}[\![\_]\!]$ 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. 6) – 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 $\mathcal{N}[\![\_]\!]$ injektiv ist.

iii'. Statt dieser Hilfslemmata kann man die Rückrichtung auch über Terminierung beweisen: Angenommen, $\mathcal{A}[\![a]\!]\,\sigma = \mathcal{N}[\![n]\!]$. Da $\sigma \vdash \langle \_ \rangle \to_{\mathsf{A}} \langle \_ \rangle$ 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 $\mathcal{N}[\![\_]\!]$ injektiv ist, gilt also $n = n'$.

Wenn $\mathcal{N}[\![\_]\!]$ 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 $\sigma \vdash \langle a \rangle \to_{\mathsf{A}} \langle n' \rangle$ und $\mathcal{N}[\![n']\!] = n$ gibt