4 Ein Compiler für While

Reale Rechner verarbeiten Assembler-Code und keine Syntaxbäume. Sprachen wie While sind damit nicht direkt auf einem solchen Rechner ausführbar, sondern müssen übersetzt werden. Die Regeln der Big-Step-Semantik (und auch der Small-Step-Semantik) lassen sich beispielsweise direkt in Prolog-Regeln konvertieren, die ein Prolog-Interpreter ausführen kann. Der für die Regeln spezialisierte Interpreter führt dann das Programm aus, übersetzt es also in eine Ausführung des Programms auf einem konkreten Rechner. Dabei wird aber das auszuführende Programm selbst nicht in eine für den Rechner geeignetere Darstellung übersetzt.

Direkter geht es, wenn man einen solchen Rechner und seine Instruktionen selbst formal modelliert und einen Übersetzer (Compiler) für While-Programme schreibt, der semantisch äquivalente Programme erzeugt. In diesem Abschnitt wird dieser Ansatz für ein sehr abstraktes Modell eines solchen Rechners für die Sprache While ausgearbeitet.

4.1 Ein abstraktes Modell eines Rechners ASM

Der abstrakte Rechner hat einen Speicher für Daten und eine Liste von Befehlen, die er abarbeitet. In unserem einfachen Modell reichen drei Assembler-Befehle (zusammengefasst in der Menge Asm), zwei zur Kontrollflusssteuerung und eine Datenoperation.

Definition 9 (Instruktionen in ASM).

`ASSN` x Aexp	Zuweisung
`JMP` k	relativer Sprung ($k \in \mathbb{Z}$)
`JMPF` k Bexp	bedingter, relativer Sprung ($k \in \mathbb{Z}$)

Ein Assembler-Programm (ASM) P besteht aus einer unveränderlichen Liste der abzuarbeitenden Befehle, angefangen mit dem ersten der Liste.

Neben dem Zustand für den Variableninhalt gibt ein *Programmzähler* an, die wievielte Instruktion der Liste die nächste ist, die abgearbeitet werden muss. Notation für einen einzelnen Ausführungsschritt: $P \vdash \langle i,\ \sigma\rangle \to \langle i',\ \sigma'\rangle$. Für ein gegebenes Programm (Instruktionsliste) P transformiert die i-te Instruktion in P den Zustand σ in den Zustand σ' und i' bezeichnet die nächste auszuführende Instruktion. P_i bezeichne das i-te Element von P und ist nur definiert, wenn i nicht negativ und kleiner als die Länge von P ist.

Definition 10 (Semantik von ASM).
Die Semantik $P \vdash \langle \_,\ \_\rangle \to \langle \_,\ \_\rangle$ eines ASM-Programms P ist durch folgende Regeln definiert.

$$\textsc{Assn}\colon \frac{P_i = \texttt{ASSN}\ x\ a}{P \vdash \langle i,\ \sigma\rangle \to \langle i+1,\ \sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma]\rangle} \qquad \textsc{Jmp}\colon \frac{P_i = \texttt{JMP}\ k}{P \vdash \langle i,\ \sigma\rangle \to \langle i+k,\ \sigma\rangle}$$

$$\textsc{JmpFT}\colon \frac{P_i = \texttt{JMPF}\ k\ b \qquad \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt}}{P \vdash \langle i,\ \sigma\rangle \to \langle i+1,\ \sigma\rangle} \qquad \textsc{JmpFF}\colon \frac{P_i = \texttt{JMPF}\ k\ b \qquad \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff}}{P \vdash \langle i,\ \sigma\rangle \to \langle i+k,\ \sigma\rangle}$$

Wenn i negativ oder größer als die Länge von P ist, ist keine Ausführung möglich.

Die gesamte Semantik eines Programms P in einem Zustand σ ist wieder über die maximalen Ableitungssequenzen von $\langle 0,\ \sigma\rangle$ gegeben; oder aber durch die blockierten Konfigurationen $\langle i',\ \sigma\rangle$, die in der transitiven Hülle $P \vdash \langle i,\ \sigma\rangle \xrightarrow{*} \langle i',\ \sigma'\rangle$ von $\langle 0,\ \sigma\rangle$ aus erreichbar sind, und die Existenz unendlicher Ausführungen $P \vdash \langle 0,\ \sigma\rangle \xrightarrow{\infty}$.

Übung: Welche Konstrukte und Regeln der Assembler-Sprache sind überflüssig und könnten durch die anderen simuliert werden?

4.2 Ein Compiler von While nach ASM

Sei $P +\!\!+ P'$ die Verkettung der beiden Listen P und P' und $|P|$ die Länge von P.

Definition 11 (Compiler). Der Compiler von While nach ASM sei durch die Funktion comp definiert:

$$\begin{aligned}
\text{comp}(\texttt{skip}) &= [] \\
\text{comp}(x\ \texttt{:=}\ a) &= [\texttt{ASSN}\ x\ a] \\
\text{comp}(c_1\texttt{;}\ c_2) &= \text{comp}(c_1) +\!\!+ \text{comp}(c_2) \\
\text{comp}(\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2) &= [\texttt{JMPF}\ k_1\ b] +\!\!+ \text{comp}(c_1) +\!\!+ [\texttt{JMP}\ k_2] +\!\!+ \text{comp}(c_2) \\
&\quad \text{wobei } k_1 = |\text{comp}(c_1)| + 2 \text{ und } k_2 = |\text{comp}(c_2)| + 1 \\
\text{comp}(\texttt{while (}b\texttt{) do}\ c) &= [\texttt{JMPF}\ (k+2)\ b] +\!\!+ \text{comp}(c) +\!\!+ [\texttt{JMP}\ -(k+1)] \\
&\quad \text{wobei } k = |\text{comp}(c)|
\end{aligned}$$

Beispiel 6.
Das Kompilat des Programms `z := 0; while (y <= x) do (z := z + 1; x := x - y)` ist:

$$[\texttt{ASSN z 0},\ \texttt{JMPF}\ 4\ (\texttt{y <= x}),\ \texttt{ASSN z (z + 1)},\ \texttt{ASSN x (x - y)},\ \texttt{JMP}\ -3]$$

Für `(if (x <= y) then x := x + y; y := x - y; x := x - y else y := x); z := 5` ergibt sich folgendes Kompilat – unabhängig von der Klammerung der Sequenz im `then`-Zweig:

$$[\texttt{JMPF}\ 5\ (\texttt{x <= y}),\ \texttt{ASSN x (x + y)},\ \texttt{ASSN y (x - y)},\ \texttt{ASSN x (x - y)},\ \texttt{JMP}\ 2,\ \texttt{ASSN y x},\ \texttt{ASSN z 5}\ ]$$

Übersetzt man `if (x <= -1) then x := -1 * x else skip`, so erhält man:

$$[\texttt{JMPF}\ 3\ (\texttt{x <= -1}),\ \texttt{ASSN x (-1 * x)},\ \texttt{JMP}\ 1]$$

4.3 Korrektheit des Compilers

Ein Compiler soll die Semantik eines Programms nicht verändern. Dadurch, dass die Semantik von While und ASM formal gegeben sind, lässt sich das auch exakt formulieren und beweisen:

- Wenn $\langle c,\ \sigma\rangle \Downarrow \sigma'$, dann $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |\text{comp}(c)|,\ \sigma'\rangle$.
- Wenn es keine Big-Step-Ableitung für $\langle c,\ \sigma\rangle$ gibt, dann $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{\infty}$.

Theorem 12 (ASM simuliert Big-Step).
Wenn $\langle c,\ \sigma\rangle \Downarrow \sigma'$, dann $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |\text{comp}(c)|,\ \sigma'\rangle$.

Dieses Theorem folgt direkt aus folgender Verallgemeinerung, die erlaubt, dass die Maschinenbefehlssequenz in beliebigen Code eingebettet ist.

Lemma 13. Seien P_1 und P_2 beliebige ASM Programme.
Wenn $\langle c,\ \sigma\rangle \Downarrow \sigma'$, dann $P_1 +\!+ \text{comp}(c) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |\text{comp}(c)|,\ \sigma'\rangle$.

Beweis. Beweis durch Regelinduktion über $\langle c,\ \sigma\rangle \Downarrow \sigma'$, P_1 und P_2 beliebig. Notation: $|c| = |\text{comp}(c)|$

- Fall SKIP_{BS}: Zu zeigen:
  Für alle P_1 und P_2 gilt $P_1 +\!+ \text{comp}(\texttt{skip}) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |\texttt{skip}|,\ \sigma\rangle$.

  Trivial wegen $|\texttt{skip}| = 0$.

- Fall ASS_{BS}: Zu zeigen: Für alle P_1 und P_2 gilt
  $P_1 +\!+ \text{comp}(x\ \texttt{:=}\ a) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |x\ \texttt{:=}\ a|,\ \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]\rangle$.

  Beweis: $P_1 +\!+ [\texttt{ASSN}\ x\ a] +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \to \langle |P_1| + 1,\ \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]\rangle$ nach Regel ASS_{N}, da $(P_1 +\!+ [\texttt{ASSN}\ x\ a] +\!+ P_2)_{|P_1|} = \texttt{ASSN}\ x\ a$.

- Fall SEQ_{BS}: Zu zeigen: Für alle P_1 und P_2 gilt
  $P_1 +\!+ \text{comp}(c_1\texttt{;}\ c_2) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |c_1\texttt{;}\ c_2|,\ \sigma''\rangle$.

  Induktionsannahmen: Für beliebige P_1 und P_2 gelten
  $P_1 +\!+ \text{comp}(c_1) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |c_1|,\ \sigma'\rangle$ und
  $P_1 +\!+ \text{comp}(c_2) +\!+ P_2 \vdash \langle |P_1|,\ \sigma'\rangle \xrightarrow{*} \langle |P_1| + |c_2|,\ \sigma''\rangle$.

  Instanziiert man in der ersten Induktionsannahme P_2 mit $\text{comp}(c_2) +\!+ P_2$ und P_1 der zweiten Induktionsannahme mit $P_1 +\!+ \text{comp}(c_1)$, so gelten:

  $$P_1 +\!+ \text{comp}(c_1) +\!+ (\text{comp}(c_2) +\!+ P_2) \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |c_1|,\ \sigma'\rangle$$
  $$(P_1 +\!+ \text{comp}(c_1)) +\!+ \text{comp}(c_2) +\!+ P_2 \vdash \langle |P_1 +\!+ \text{comp}(c_1)|,\ \sigma'\rangle \xrightarrow{*} \langle |P_1 +\!+ \text{comp}(c_1)| + |c_2|,\ \sigma''\rangle$$

  Ausrechnen und Transitivität von $\xrightarrow{*}$ liefern die Behauptung.

- Fall IFTT_{BS}: Zu zeigen: Für alle P_1 und P_2 gilt $P_1 +\!+ \text{comp}(\texttt{if}\ (b)\ \texttt{then}\ c_1\ \texttt{else}\ c_2) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |\texttt{if}\ (b)\ \texttt{then}\ c_1\ \texttt{else}\ c_2|,\ \sigma'\rangle$.

  Induktionsannahmen:
  $\mathcal{B}[\![b]\!]\,\sigma = \texttt{tt}$ und für beliebige P_1 und P_2 gilt $P_1 +\!+ \text{comp}(c_1) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |c_1|,\ \sigma'\rangle$.

  Beweis mit der Induktionsannahme, bei der P_1 als $P_1 +\!+ [\texttt{JMPF}\ (|c_1| + 2)\ b]$ und
  P_2 als $[\texttt{JMP}\ (|c_2| + 1)] +\!+ \text{comp}(c_2) +\!+ P_2$ instanziiert werden:

  $$P_1 +\!+ [\texttt{JMPF}\ (|c_1| + 2)\ b] +\!+ \text{comp}(c_1) +\!+ [\texttt{JMP}\ (|c_2| + 1)] +\!+ \text{comp}(c_2) +\!+ P_2 \vdash$$
  $$\langle |P_1|,\ \sigma\rangle \to \langle |P_1| + 1,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + 1 + |c_1|,\ \sigma'\rangle \to \langle |P_1| + 2 + |c_1| + |c_2|,\ \sigma'\rangle$$

- Fall IFFF_{BS}: Analog zu IFTT_{BS}.

- Fall $\text{WHILETT}_{\text{BS}}$: Zu zeigen: Für alle P_1 und P_2 gilt
  $P_1 +\!+ \text{comp}(\texttt{while}\ (b)\ \texttt{do}\ c) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |\texttt{while}\ (b)\ \texttt{do}\ c|,\ \sigma''\rangle$.

  Induktionsannahmen: $\mathcal{B}[\![b]\!]\,\sigma = \texttt{tt}$ und für beliebige P_1 und P_2 gelten
  $P_1 +\!+ \text{comp}(c) +\!+ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |c|,\ \sigma'\rangle$ und $P_1 +\!+ \text{comp}(\texttt{while}\ (b)\ \texttt{do}\ c) +\!+ P_2 \vdash \langle |P_1|,\ \sigma'\rangle \xrightarrow{*} \langle |P_1| + |\texttt{while}\ (b)\ \texttt{do}\ c|,\ \sigma''\rangle$.

  Beweis mit entsprechend instanziierten Induktionshypothesen und Regel JMPFT:

  $$P_1 +\!+ [\texttt{JMPF}\ (|c| + 2)\ b] +\!+ \text{comp}(c) +\!+ [\texttt{JMP}\ -(|c| + 1)] +\!+ P_2 \vdash$$
  $$\langle |P_1|,\ \sigma\rangle \to \langle |P_1| + 1,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + 1 + |c|,\ \sigma'\rangle \to \langle |P_1|,\ \sigma'\rangle \xrightarrow{*} \langle |P_1| + |\texttt{while}\ (b)\ \texttt{do}\ c|,\ \sigma''\rangle$$

- Fall $\text{WHILEFF}_{\text{BS}}$: Induktionsannahme: $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$.
  Zu zeigen: Für alle P_1 und P_2 gilt
  $P_1 ++ \text{comp}(\texttt{while (}b\texttt{) do }c) ++ P_2 \vdash \langle |P_1|,\ \sigma\rangle \xrightarrow{*} \langle |P_1| + |\texttt{while (}b\texttt{) do }c|,\ \sigma\rangle$.
  Beweis mit Regel JMPFF:

$$P_1 ++ [\texttt{JMPF } b\ (|c|+2)] ++ \text{comp}(c) ++ [\texttt{JMP } -(|c|+1)] ++ P_2\ \vdash \langle |P_1|,\ \sigma\rangle \to \langle |P_1| + |c| + 2,\ \sigma\rangle$$

□

Alternativ lässt sich Theorem 12 auch direkt per Induktion beweisen, braucht dann aber folgendes Hilfslemma, das es erlaubt, Instruktionslisten vorne oder hinten zu erweitern:

Lemma 14 (Verschiebungslemma).

(i) Wenn $P \vdash \langle i,\ \sigma\rangle \to \langle i',\ \sigma'\rangle$, dann gilt auch, dass $P' ++ P \vdash \langle i + |P'|,\ \sigma\rangle \to \langle i' + |P'|,\ \sigma'\rangle$ und $P ++ P' \vdash \langle i,\ \sigma\rangle \to \langle i',\ \sigma'\rangle$.

(ii) Wenn $P \vdash \langle i,\ \sigma\rangle \xrightarrow{n} \langle i',\ \sigma'\rangle$, dann gilt auch, dass $P' ++ P \vdash \langle i + |P'|,\ \sigma\rangle \xrightarrow{n} \langle i' + |P'|,\ \sigma'\rangle$ und $P ++ P' \vdash \langle i,\ \sigma\rangle \xrightarrow{n} \langle i',\ \sigma'\rangle$.

Beweis. (i) Fallunterscheidung nach P_i. (ii) Induktion über n, Induktionsschritt mit (i). □

Direkter Beweis von Theorem 12: Beweis durch Regelinduktion über $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Notation $|c| = |\text{comp}(c)|$.

- Fall SKIP_{BS}: Zu zeigen: $\text{comp}(\texttt{skip}) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |\texttt{skip}|,\ \sigma\rangle$. Trivial wegen $|\texttt{skip}| = 0$.
- Fall ASS_{BS}: Zu zeigen: $\text{comp}(x\ \texttt{:=}\ a) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |x\ \texttt{:=}\ a|,\ \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]\rangle$.
  Beweis: $[\texttt{ASSN } x\ a] \vdash \langle 0,\ \sigma\rangle \to \langle 1,\ \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]\rangle$ nach Regel ASSN.
- Fall SEQ_{BS}: Zu zeigen: $\text{comp}(c_1\texttt{;}\ c_2) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |c_1\texttt{;}\ c_2|,\ \sigma''\rangle$.
  Induktionsannahmen: $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |c_1|,\ \sigma'\rangle$ und $\text{comp}(c_2) \vdash \langle 0,\ \sigma'\rangle \xrightarrow{*} \langle |c_2|,\ \sigma''\rangle$.
  Mit dem Verschiebungslemma 14 folgt aus den Induktionsannahmen, dass

$$\text{comp}(c_1) ++ \text{comp}(c_2) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |c_1|,\ \sigma'\rangle$$
$$\text{comp}(c_1) ++ \text{comp}(c_2) \vdash \langle 0 + |c_1|,\ \sigma'\rangle \xrightarrow{*} \langle |c_2| + |c_1|,\ \sigma''\rangle$$

  Transitivität von $\xrightarrow{*}$ liefert die Behauptung.
- Fall IFTT_{BS}: Induktionsannahmen: $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |c_1|,\ \sigma'\rangle$ und $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$.
  Zu zeigen: $\text{comp}(\texttt{if (}b\texttt{) then }c_1\ \texttt{else}\ c_2) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |\texttt{if (}b\texttt{) then }c_1\ \texttt{else}\ c_2|,\ \sigma'\rangle$.
  Beweis mit dem Verschiebungslemma 14:

$$[\texttt{JMPF } (|c_1|+2)\ b] ++ \text{comp}(c_1) ++ [\texttt{JMP } (|c_2|+1)] ++ \text{comp}(c_2)\ \vdash$$
$$\langle 0,\ \sigma\rangle \to \langle 1,\ \sigma\rangle \xrightarrow{*} \langle 1 + |c_1|,\ \sigma'\rangle \to \langle 2 + |c_1| + |c_2|,\ \sigma'\rangle$$

- Fall IFFF_{BS}: Analog zu IFTT_{BS}.
- Fall $\text{WHILETT}_{\text{BS}}$: Induktionsannahmen: $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$, $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |c|,\ \sigma'\rangle$ und $\text{comp}(\texttt{while (}b\texttt{) do }c) \vdash \langle 0,\ \sigma'\rangle \xrightarrow{*} \langle |\texttt{while (}b\texttt{) do }c|,\ \sigma''\rangle$.
  Zu zeigen: $\text{comp}(\texttt{while (}b\texttt{) do }c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle |\texttt{while (}b\texttt{) do }c|,\ \sigma''\rangle$.
  Beweis mit dem Verschiebungslemma 14:

$$[\texttt{JMPF } (|c|+2)\ b] ++ \text{comp}(c) ++ [\texttt{JMP } -(|c|+1)]\ \vdash$$
$$\langle 0,\ \sigma\rangle \to \langle 1,\ \sigma\rangle \xrightarrow{*} \langle |c|+1,\ \sigma'\rangle \to \langle 0,\ \sigma'\rangle \xrightarrow{*} \langle |\texttt{while (}b\texttt{) do }c|,\ \sigma''\rangle$$

- Fall WHILEFF$_{\text{BS}}$: Induktionsannahme: $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}$.
  Zu zeigen: comp(`while (`b`) do `c) $\vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\texttt{while (}b\texttt{) do }c|,\ \sigma\rangle$.
  Beweis mit Regel JMPFF:

$$[\texttt{JMPF}\ b\ (|c|+2)] +\!\!+ \text{comp}(c) +\!\!+ [\texttt{JMP}\ -(|c|+1)] \vdash \langle 0,\ \sigma\rangle \to \langle |c|+2,\ \sigma\rangle \qquad \square$$

Dieses Theorem zeigt, dass es zu jeder terminierenden Ausführung in der Big-Step-Semantik eine entsprechende (terminierende) Ausführung des übersetzten Programms gibt. Das Theorem sagt jedoch nichts über nicht terminierende Programme aus: Wir haben den zweiten Teil des obigen Korrektheitsbegriffs – nicht terminierende Ausführungen in While terminieren auch in ASM nicht – noch nicht bewiesen. Da Nichttermination aber in der Big-Step-Semantik nicht direkt ausdrückbar ist, wäre ein direkter Beweis formal sehr aufwändig. Stattdessen zeigt man üblicherweise die umgekehrte Simulationsrichtung: Wenn $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c)|,\ \sigma'\rangle$, dann $\langle c,\ \sigma\rangle \Downarrow \sigma'$.

Für diesen Beweis benötigen wir noch den Begriff der Abgeschlossenheit. Das Verschiebungslemma erlaubte uns, beliebige Instruktionslisten zusammenzufügen. Für die Rückrichtung müssen wir Instruktionslisten an geeigneten Stellen aufteilen können. Dies ist aber nur möglich, wenn es keine Sprünge über die Aufteilungsgrenze gibt. Abgeschlossenheit ist ein ausreichendes Kriterium dafür:

Definition 12 (Abgeschlossenheit). Eine Instruktionsliste P heißt *abgeschlossen*, wenn alle Sprünge in P nur an die (absoluten) Positionen aus $\{0, \ldots, |P|\}$ gehen. Formal: Für alle $i < |P|$ mit $P_i = \texttt{JMP}\ n$ oder $P_i = \texttt{JMPF}\ n\ b$ gilt: $0 \le i + n \le |P|$.

Beispiel 7.
[`JMPF 3 (x <= -1), ASSN x (-1 * x), JMP 1`] ist abgeschlossen, nicht aber [`JMPF 3 (x <= 5), ASSN x 17`].

Lemma 15. comp(c) ist abgeschlossen.

Beweis. Induktion über c. $\square$

Lemma 16 (Aufteilungslemma). Sei P abgeschlossen, $0 \le i \le |P|$ und $j \notin \{|P_0|, \ldots, |P_0| + |P| - 1\}$. Wenn $P_0 +\!\!+ P +\!\!+ P_1 \vdash \langle |P_0| + i,\ \sigma\rangle \overset{n}{\to} \langle j,\ \sigma'\rangle$, dann gibt es σ^*, n_1 und n_2 mit $n = n_1 + n_2$, $P \vdash \langle i,\ \sigma\rangle \overset{n_1}{\to} \langle |P|,\ \sigma^*\rangle$ und $P_0 +\!\!+ P +\!\!+ P_1 \vdash \langle |P_0| + |P|,\ \sigma^*\rangle \overset{n_2}{\to} \langle j,\ \sigma'\rangle$.

Beweis. Abkürzungen: $J = \{|P_0|, \ldots, |P_0| + |P| - 1\}$, $Q = P_0 +\!\!+ P +\!\!+ P_1$.
Induktion über n (σ und i beliebig):

- Basisfall $n = 0$: Zu zeigen: Wenn (i) $Q \vdash \langle |P_0| + i,\ \sigma\rangle \overset{0}{\to} \langle j,\ \sigma'\rangle$ und (ii) $0 \le i < |P|$, dann gibt es σ^*, n_1 und n_2 mit $0 = n_1 + n_2$, $P \vdash \langle i,\ \sigma\rangle \overset{n_1}{\to} \langle |P|,\ \sigma^*\rangle$ und $Q \vdash \langle |P_0| + |P|,\ \sigma^*\rangle \overset{n_2}{\to} \langle j,\ \sigma'\rangle$.
  Aus (i) folgt $j = |P_0| + i$ und $\sigma' = \sigma$. Mit (ii) und $j \notin J$ folgt $i = |P|$. Damit folgt die Behauptung mit $n_1 = 0$, $n_2 = 0$ und $\sigma^* = \sigma$.
- Induktionsschritt $n + 1$:
  Induktionsannahme: Für alle $0 \le i \le |P|$ und σ gilt: Wenn $Q \vdash \langle |P_0| + i,\ \sigma\rangle \overset{n}{\to} \langle j,\ \sigma'\rangle$, dann gibt es σ^*, n_1 und n_2 mit $n = n_1 + n_2$, $P \vdash \langle i,\ \sigma\rangle \overset{n_1}{\to} \langle |P|,\ \sigma^*\rangle$ und $Q \vdash \langle |P_0| + |P|,\ \sigma^*\rangle \overset{n_2}{\to} \langle j,\ \sigma'\rangle$.
  Zu zeigen: Wenn (i) $Q \vdash \langle |P_0| + i,\ \sigma\rangle \overset{n+1}{\to} \langle j,\ \sigma'\rangle$ und $0 \le i \le |P|$, dann gibt es σ^*, n_1 und n_2 mit $n + 1 = n_1 + n_2$, $P \vdash \langle i,\ \sigma\rangle \overset{n_1}{\to} \langle |P|,\ \sigma^*\rangle$ und $Q \vdash \langle |P_0| + |P|,\ \sigma^*\rangle \overset{n_2}{\to} \langle j,\ \sigma'\rangle$.
  Wenn $i = |P|$, dann wähle $n_1 = 0$, $n_2 = n + 1$ und $\sigma^* = \sigma$.
  Sei also o.B.d.A (ii) $0 \le i < |P|$. Aus (i) gibt es i' und σ'' mit $Q \vdash \langle |P_0| + i,\ \sigma\rangle \to \langle i',\ \sigma''\rangle \overset{n}{\to} \langle j,\ \sigma\rangle$. Aus $Q \vdash \langle |P_0| + i,\ \sigma\rangle \to \langle i',\ \sigma''\rangle$ folgt (iii) $|P_0| \le i' \le |P_0| + |P|$ und (iv) $P \vdash \langle i,\ \sigma\rangle \to \langle i' - |P_0|,\ \sigma''\rangle$ durch Fallunterscheidung:
  - Fall ASSN: Damit $Q_{|P_0|+i} = \texttt{ASSN}\ x\ a$, $i' = |P_0| + i + 1$, $\sigma'' = \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]$. Also $P_i = \texttt{ASSN}\ x\ a$ und $P \vdash \langle i,\ \sigma\rangle \to \langle i' - |P_0|,\ \sigma''\rangle$ nach Regel ASSN. Mit (ii) folgt $|P_0| \le i' \le |P_0| + |P|$.

– Fall JMP: Damit $Q_{|P_0|+i} = \texttt{JMP}\ k = P_i$, $i' = |P_0| + i + k$, $\sigma'' = \sigma$. Somit $P \vdash \langle i,\ \sigma\rangle \to \langle i+k,\ \sigma''\rangle$ nach Regel JMP. Da P abgeschlossen ist, gilt $0 \le i + k \le |P|$, somit $|P_0| \le i' \le |P_0| + |P|$.
– Fall JMPFT: Analog zu Fall ASSN.
– Fall JMPFF: Analog zu Fall JMP.

Beweis des Induktionsfalls durch Fallunterscheidung:
– Fall $i' = |P_0| + |P|$: Setze $n_1 = 1$, $n_2 = n$, $\sigma^* = \sigma''$. Aus (iv) folgt $P \vdash \langle i,\ \sigma\rangle \to \langle |P|,\ \sigma''\rangle$. Mit $Q \vdash \langle i',\ \sigma''\rangle \overset{n}{\to} \langle j,\ \sigma\rangle$ folgt die Behauptung.
– Fall $|P_0| \le i' < |P_0| + |P|$: Somit $0 \le i' - |P_0| < |P|$. Nach der Induktionsannahme für $i = i' - |P_0|$ und $\sigma = \sigma''$ und $Q \vdash \langle i',\ \sigma''\rangle \overset{n}{\to} \langle j,\ \sigma'\rangle$ gibt es σ^*, n_1 und n_2 mit $n = n_1 + n_2$, (v) $P \vdash \langle i' - |P_0|,\ \sigma\rangle \overset{n_1}{\to} \langle |P|,\ \sigma^*\rangle$ und (vi) $Q \vdash \langle |P_0| + |P|,\ \sigma^*\rangle \overset{n_2}{\to} \langle j,\ \sigma'\rangle$. Aus (iv) und (v) folgt $P \vdash \langle i,\ \sigma\rangle \overset{n_1+1}{\to} \langle |P|,\ \sigma^*\rangle$. Mit (vi) und $n + 1 = (n_1 + 1) + n_2$ folgt die Behauptung. □

Theorem 17 (Big-Step simuliert ASM).
Wenn $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c)|,\ \sigma'\rangle$, dann $\langle c,\ \sigma\rangle \Downarrow \sigma'$.

Beweis. Beweis durch Induktion über c (σ, σ' beliebig):

- Fall `skip`: Zu zeigen: Wenn (i) $[] \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle 0,\ \sigma'\rangle$, dann $\langle \texttt{skip},\ \sigma\rangle \Downarrow \sigma'$.
  Aus (i) folgt durch Regelinversion, dass $\sigma' = \sigma$, damit die Behauptung mit Regel SKIP_{BS}.
- Fall x `:=` a: Zu zeigen: Wenn (i) $[\texttt{ASSN}\ x\ a] \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle 1,\ \sigma'\rangle$, dann $\langle x\ \texttt{:=}\ a,\ \sigma\rangle \Downarrow \sigma'$.
  Aus (i) folgt durch Regelinversion, dass $\sigma' = \sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]$. Damit gilt $\langle x\ \texttt{:=}\ a,\ \sigma\rangle \Downarrow \sigma'$ nach Regel ASS_{BS}.
- Fall c_1 `;` c_2: Abkürzungen: $P = \text{comp}(c_1) +\!+ \text{comp}(c_2)$ und $l = |\text{comp}(c_1)| + |\text{comp}(c_2)|$
  Induktionsannahmen: Für alle σ und σ' gilt: Wenn $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c_1)|,\ \sigma'\rangle$, dann $\langle c_1,\ \sigma\rangle \Downarrow \sigma'$; und wenn $\text{comp}(c_2) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c_2)|,\ \sigma'\rangle$, dann $\langle c_2,\ \sigma\rangle \Downarrow \sigma'$.
  Zu zeigen: Wenn (i) $P \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle l,\ \sigma'\rangle$, dann $\langle c_1\ \texttt{;}\ c_2,\ \sigma\rangle \Downarrow \sigma'$.

  Nach Lem. 15 sind $\text{comp}(c_1)$ und $\text{comp}(c_2)$ abgeschlossen. Nach dem Aufteilungslemma 16 und (i) gibt es ein σ^* mit (ii) $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c_1)|,\ \sigma^*\rangle$ und (iii) $P \vdash \langle |\text{comp}(c_1)|,\ \sigma^*\rangle \overset{*}{\to} \langle l,\ \sigma'\rangle$.

  Aus (ii) folgt mit der Induktionsannahme, dass $\langle c_1,\ \sigma\rangle \Downarrow \sigma^*$.

  Wegen (iii) gibt es nach dem Aufteilungslemma ein σ^{**} mit (iv) $\text{comp}(c_2) \vdash \langle 0,\ \sigma^*\rangle \overset{*}{\to} \langle |\text{comp}(c_2)|,\ \sigma^{**}\rangle$ und (v) $P \vdash \langle l,\ \sigma^{**}\rangle \overset{*}{\to} \langle l,\ \sigma'\rangle$.

  Aus (iv) folgt mit der Induktionsannahme, dass $\langle c_2,\ \sigma^*\rangle \Downarrow \sigma^{**}$.

  Aus (v) folgt durch Regelinversion, dass $\sigma^{**} = \sigma'$ und damit die Behauptung mit Regel SEQ_{BS}.
- Fall `if (`b`) then` c_1 `else` c_2: Abkürzungen: $l_1 = |\text{comp}(c_1)|$, $l_2 = |\text{comp}(c_2)|$,
  $P = [\texttt{JMPF}\ (l_1 + 2)\ b] +\!+ \text{comp}(c_1) +\!+ [\texttt{JMP}\ (l_2 + 1)] +\!+ \text{comp}(c_2)$.
  Induktionsannahmen: Für alle σ und σ' gilt: Wenn $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c_1)|,\ \sigma'\rangle$, dann $\langle c_1,\ \sigma\rangle \Downarrow \sigma'$; und wenn $\text{comp}(c_2) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |\text{comp}(c_2)|,\ \sigma'\rangle$, dann $\langle c_2,\ \sigma\rangle \Downarrow \sigma'$.
  Zu zeigen: Wenn (i) $P \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |l_1 + l_2 + 2|,\ \sigma'\rangle$, dann $\langle \texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2,\ \sigma\rangle \Downarrow \sigma'$.

  Beweis: Fallunterscheidung nach $\mathcal{B}[\![b]\!]\,\sigma$.
  – Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt}$: Aus (i) folgt $P \vdash \langle 0,\ \sigma\rangle \to \langle 1,\ \sigma\rangle \overset{*}{\to} \langle l_1 + l_2 + 2,\ \sigma'\rangle$ durch Regelinversion. Da $\text{comp}(c_1)$ abgeschlossen ist (Lem. 15), gibt es mit dem Aufteilungslemma 16 ein σ^* mit (i) $\text{comp}(c_1) \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle l_1,\ \sigma^*\rangle$ und (ii) $P \vdash \langle l_1 + 1,\ \sigma^*\rangle \overset{*}{\to} \langle l_1 + l_2 + 2,\ \sigma'\rangle$. Aus (i) folgt mit der Induktionsannahme, dass $\langle c_1,\ \sigma\rangle \Downarrow \sigma^*$.
    Aus (ii) und $P_{l_1+1} = \texttt{JMP}\ (l_2 + 1)$ folgt mit Regel-Inversion, dass $\sigma^* = \sigma'$. Zusammen mit $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt}$ und $\langle c_1,\ \sigma\rangle \Downarrow \sigma^*$ folgt $\langle \texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2,\ \sigma\rangle \Downarrow \sigma'$ nach Regel IFTT_{BS}.
  – Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}$: Analog für c_2 statt c_1.

- Fall `while (b) do c`: Abkürzungen: $P = \text{comp}(\texttt{while (b) do } c)$ und $l = |\text{comp}(c)|$.
  Induktionsannahme I: Wenn $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle l,\ \sigma'\rangle$, dann $\langle c,\ \sigma\rangle \Downarrow \sigma'$ für beliebige σ, σ'.
  Zu zeigen: Wenn $P \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle l+2,\ \sigma'\rangle$, dann $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$.

  Beweis einer stärkeren Aussage durch vollständige Induktion über n (σ beliebig):
  Wenn $P \vdash \langle 0,\ \sigma\rangle \xrightarrow{n} \langle l+2,\ \sigma'\rangle$, dann $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$.

  Sei n beliebig. Induktionsannahme II:
  Für alle $m < n$ und σ gilt: Wenn $P \vdash \langle 0,\ \sigma\rangle \xrightarrow{m} \langle l+2,\ \sigma'\rangle$, dann $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$.
  Zu zeigen: Wenn (i) $P \vdash \langle 0,\ \sigma\rangle \xrightarrow{n} \langle l+2,\ \sigma'\rangle$, dann $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$.

  Fallunterscheidung nach $\mathcal{B}[\![b]\!]\sigma$:
  - Fall $\mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$: Aus (i) folgt dann $n = 1$, $\sigma = \sigma'$ durch Regelinversion. Nach Regel $\text{WHILEFF}_{\text{BS}}$ gilt $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$.
  - Fall $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$: Aus (i) folgt damit $n > 0$ und $P \vdash \langle 0,\ \sigma\rangle \to \langle 1,\ \sigma\rangle \xrightarrow{n-1} \langle l+2,\ \sigma'\rangle$.
    Da $\text{comp}(c)$ abgeschlossen ist (Lem. 15), gibt es nach dem Aufteilungslemma 16 σ^*, n_1 und n_2 mit $n - 1 = n_1 + n_2$, (ii) $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{n_1} \langle l,\ \sigma^*\rangle$ und (iii) $P \vdash \langle l+1,\ \sigma^*\rangle \xrightarrow{n_2} \langle l+2,\ \sigma'\rangle$.
    Aus (ii) folgt mit Induktionsannahme I, dass $\langle c,\ \sigma\rangle \Downarrow \sigma^*$.

    Aus (iii) und $P_{l+1} = \texttt{JMP } -(l+1)$ folgt durch Regelinversion: $P \vdash \langle l+1,\ \sigma^*\rangle \to \langle 0,\ \sigma^*\rangle \xrightarrow{n_2-1} \langle l+2,\ \sigma'\rangle$.

    Aus $P \vdash \langle 0,\ \sigma^*\rangle \xrightarrow{n_2-1} \langle l+2,\ \sigma'\rangle$ folgt mit der Induktionsannahme II für $m = n_2 - 1 < n$ und $\sigma = \sigma^*$, dass $\langle \texttt{while (b) do } c,\ \sigma^*\rangle \Downarrow \sigma'$.
    Zusammen mit $\langle c,\ \sigma\rangle \Downarrow \sigma^*$ und $\mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$ folgt $\langle \texttt{while (b) do } c,\ \sigma\rangle \Downarrow \sigma'$ mit Regel $\text{WHILETT}_{\text{BS}}$. □

Aus diesem Theorem lässt sich nun „einfach“ die zweite Korrektheitsaussage des Compilers beweisen. Zuerst aber noch folgendes einfaches Hilfslemma:

Lemma 18. Sei P abgeschlossen und $0 \le i < |P|$.

(i) Wenn $P \vdash \langle i,\ \sigma\rangle \to \langle i',\ \sigma'\rangle$, dann $0 \le i' \le |P|$.

(ii) Wenn $P \vdash \langle i,\ \sigma\rangle \xrightarrow{*} \langle i',\ \sigma'\rangle$ und $0 \le i \le |P|$, dann $0 \le i' \le |P|$.

Beweis. (i) durch Regelinversion, (ii) durch Induktion über $\xrightarrow{*}$, Induktionsschritt mit (i). □

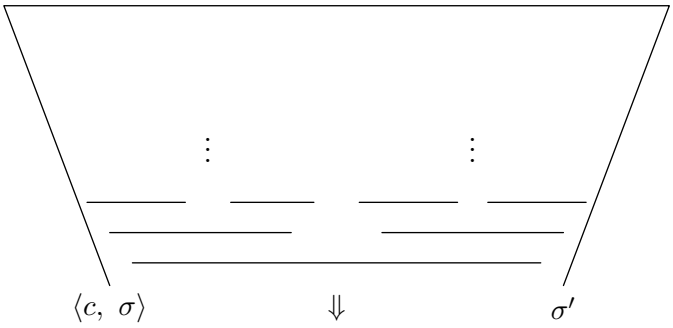
Korollar 19. Gibt es kein σ' mit $\langle c,\ \sigma\rangle \Downarrow \sigma'$, dann $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{\infty}$.

Beweis. Beweis durch Widerspruch. Angenommen, $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \not\xrightarrow{\infty}$. Dann gibt es i' und σ' mit $\text{comp}(c) \vdash \langle 0,\ \sigma\rangle \xrightarrow{*} \langle i',\ \sigma'\rangle \not\to$ Da $\text{comp}(c)$ abgeschlossen ist (Lem. 15), folgt mit Lem. 18, dass $0 \le i' \le |\text{comp}(c)|$. Da $\langle i',\ \sigma'\rangle$ blockiert ist, gilt $i' = |\text{comp}(c)|$. Nach Thm. 17 gilt also $\langle c,\ \sigma\rangle \Downarrow \sigma'$. Widerspruch zur Voraussetzung. □

4.4 Vergleich der verschiedenen Semantiken

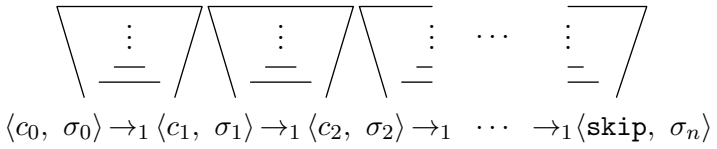
Big-Step-Semantik

- Das gesamte Programm wertet in einem Schritt zu einem Endzustand aus.
- Alle Berechnungsdetails sind in dem Ableitungsbaum versteckt.
- Beweisprinzip: Regelinduktion über die Auswertungsrelation.
- Nichttermination und Blockieren sind nicht unterscheidbar.



Small-Step-Semantik

- Ableitungsfolge enthält explizit alle Berechnungsschritte.
- Jeder Schritt wird durch einen einzelnen Ableitungsbaum gerechtfertigt.



- Beweisprinzip: Regelinduktion für einzelne Schritte und Induktion über die Länge der Ableitungsfolge
- Nichttermination und Blockieren ausdrückbar
- Die transitive, reflexive Hülle abstrahiert von den störenden, expliziten Zwischenschritten.

ASM $P \vdash \langle i_0,\ \sigma_0\rangle \to \langle i_1,\ \sigma_1\rangle \to \langle i_2,\ \sigma_2\rangle \to \ldots \to \langle |P|,\ \sigma_n\rangle$

- Programm ist eine Liste, hat keine Baumstruktur.
- Ableitungsfolgen beschreiben wie bei einer Small-Step-Semantik die einzelnen Berechnungsschritte
- Jeder Schritt wird nur durch eine einzelne Regel gerechtfertigt.
- Beweisprinzip: Fallunterscheidung über die Instruktionen und Induktion über die Länge der Ableitungsfolge
- Nichttermination und Blockieren ausdrückbar.