Kapitel 6

Programme mit Schleifen

Wir betrachten jetzt eine einfache imperative Programmiersprache IMP. IMP verfügt über zuweisbare Variablen, Konditionale und Schleifen, hat aber keine Prozeduren. IMP verfügt nur über den Typ `int`.

Wir werden zwei Semantiken für IMP definieren, die als denotationale und operationale Semantik bezeichnet werden. Die denotationale Semantik wird nach dem bereits von der Aussagenlogik her bekannten Muster definiert und ordnet jedem Objekt der abstrakten Syntax eine Denotation zu. Interessant ist die Denotation von Schleifen, die wir mittels eines Fixpunktoperators definieren werden.

Die operationale Semantik von IMP werden wir relational mithilfe von Inferenzregeln definieren. Die Definition der operationalen Semantik kommt ohne die Verwendung eines Fixpunktoperators aus und ist daher für mathematisch unerfahrene Leser leichter zu verstehen.

Wir werden beweisen, dass die denotationale und die operationale Semantik von IMP übereinstimmen. Dabei kommen strukturelle Induktion, Regelinduktion und natürliche Induktion zum Einsatz.

Lesematerial

Unsere Sprache IMP entspricht im Wesentlichen der Sprache IMP in [Winskel], wir haben lediglich auf einige unwesentliche Sprachkonstrukte verzichtet. Da wir die mathematischen Grundlagen bereits gelegt haben, beginnen wir sofort mit der Definition der denotationalen Semantik. Aus demselben Grund arbeiten wir statt mit partiellen Funktionen $\Sigma \rightharpoonup \Sigma$ sofort mit totalen Funktionen $\Sigma \to \Sigma_\perp$.

[Winskel, Kapitel 2 und 5]

$$
\begin{array}{lllll}
n & \in & Con = \mathbb{Z} & & \textit{Konstante} \\
X & \in & Loc & & \textit{Lokation} \\
a & \in & Aexp = n \mid X \mid a_1 + a_2 & & \textit{Arithmetischer Ausdruck} \\
b & \in & Bexp = a_1 \leq a_2 & & \textit{Boolscher Ausdruck} \\
c & \in & Com = & & \textit{Kommando} \\
 & & \quad X := a & & \quad \textit{Zuweisung} \\
 & & \mid c_1; c_2 & & \quad \textit{Sequentialisierung} \\
 & & \mid \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 & & \quad \textit{Konditional} \\
 & & \mid \texttt{while}\ b\ \texttt{do}\ c & & \quad \textit{Schleife}
\end{array}
$$

Abbildung 6.1: Syntax von IMP

6.1 Syntax und denotationale Semantik von IMP

Abbildung 6.1 definiert die Syntax von IMP (relativ zu einer gegebenen, nicht-leeren Menge *Loc*). Die Lokationen von IMP bezeichnet man auch als Variablen, und die Kommandos als Anweisungen. Sie sollten sich IMP als eine Teilsprache von C oder Java vorstellen. Um die folgenden Definitionen und Beweise einfach zu halten, haben wir IMP nur mit einem Minimum an Konstrukten ausgestattet.

Sei $X_0 \in Loc$. Wir definieren die folgenden syntaktischen Abkürzungen für IMP:

$$
\begin{array}{rcl}
\texttt{false} & \mapsto & 1 \leq 0 \\
\texttt{true} & \mapsto & 1 \leq 1 \\
\texttt{skip} & \mapsto & X_0 := X_0
\end{array}
$$

Mit Schleifen können wir divergierende Programme schreiben, zum Beispiel:

```
while true do skip
```

Für die Semantik von IMP benötigen wir Funktionen

$$\sigma \in \Sigma \stackrel{\text{def}}{=} Loc \to \mathbb{Z} \qquad \textit{Zustand}$$

die Lokationen auf Zahlen abbilden und die wir als *Zustände* bezeichnen.[1]

[1] Bisher haben wir solche Funktionen als Belegungen bezeichnet.

Sei $\perp$ ein einmal gewähltes Objekt, das kein Element von Σ ist (zum Beispiel die leere Menge). Wir definieren:

$$\Sigma_\perp \overset{\text{def}}{=} \Sigma \cup \perp$$

Um ein Kommando c auszuführen, benötigen wir einen *Anfangszustand*. Wenn die Ausführung des Kommandos terminiert, bekommen wir einen *Endzustand*. Der Effekt einer terminierenden Ausführung besteht also darin, einen Anfangszustand in einen Endzustand zu überführen. Wir können also jedem Kommando c eine Funktion

$$\phi \in \Sigma \to \Sigma_\perp$$

zuordnen. Dabei bedeutet $\phi\sigma = \perp$, dass die Ausführung von c mit dem Anfangszustand σ divergiert, und $\phi\sigma = \sigma'$, dass die Ausführung von c mit dem Anfangszustand σ mit dem Endzustand σ' terminiert.

Wir wählen die gerade beschriebene Funktion als Denotation eines Kommandos. Die Denotation eines Kommandos beschreibt sein Ein/Ausgabe-Verhalten. Die Denotation enthält weniger Information als das Kommando, da sie keinen Algorithmus für die Berechnung des Endzustands vorgibt.

Abbildung 6.2 definiert die denotationale Semantik von IMP. Für jede der drei Phrasenklassen wird eine eigene Denotationsfunktion mithilfe von struktureller Rekursion definiert. Bis auf die Gleichung für Schleifen sollten Ihnen alle Gleichungen unmittelbar klar sein.

Wir erklären jetzt die Gleichung für Schleifen. Sie hat die Aufgabe, die Denotation einer Schleife

$$\texttt{while}\ b\ \texttt{do}\ c$$

mithilfe der Denotationen $\mathcal{B}[\![b]\!]$ und $\mathcal{C}[\![c]\!]$ zu definieren. Die Denotation der Schleife können wir durch die folgende, rekursiv definierte Funktion beschreiben:

$$\psi \in \Sigma \to \Sigma_\perp$$

$$\begin{aligned}\psi(\sigma) &= \texttt{if}\ \mathcal{B}[\![b]\!]\sigma = 0\ \texttt{then}\ \sigma \\ &\quad\ \texttt{else if}\ \mathcal{C}[\![c]\!]\sigma = \perp\ \texttt{then}\ \perp\ \texttt{else}\ \psi(\mathcal{C}[\![c]\!]\sigma)\end{aligned}$$

Wenn wir die Rekursion ordnungsgemäß mithilfe des ensprechenden Funktionals und des Fixpunktoperators für die VPO $\Sigma \to \Sigma_\perp$ beschreiben, erhalten wir die in Abbildung 6.2 gezeigte Hilfsfunktion Γ und die Gleichung für Schleifen. Das geübte Auge sieht auch sofort, dass $\Gamma(\beta, \phi)$ für alle zulässigen β und ϕ eine stetige Funktion ist.

$$\mathcal{A} \in Aexp \to (\Sigma \to \mathbb{Z})$$

$$\begin{aligned}
\mathcal{A}[\![n]\!] &= \lambda\,\sigma\,.\,n \\
\mathcal{A}[\![X]\!] &= \lambda\,\sigma\,.\,\sigma(X) \\
\mathcal{A}[\![a_1 + a_2]\!] &= \lambda\,\sigma\,.\,\mathcal{A}[\![a_1]\!]\sigma + \mathcal{A}[\![a_2]\!]\sigma
\end{aligned}$$

$$\mathcal{B} \in Bexp \to (\Sigma \to \mathbb{B})$$

$$\mathcal{A}[\![a_1 \leq a_2]\!] = \lambda\,\sigma\,.\,\text{if } \mathcal{A}[\![a_1]\!]\sigma \leq \mathcal{A}[\![a_2]\!]\sigma \text{ then } 1 \text{ else } 0$$

$$\mathcal{C} \in Com \to (\Sigma \to \Sigma_\perp)$$

$$\begin{aligned}
\mathcal{C}[\![X := a]\!] &= \lambda\,\sigma\,.\,\sigma[\mathcal{A}[\![a]\!]\sigma/X] \\
\mathcal{C}[\![c_1; c_2]\!] &= \lambda\,\sigma\,.\,\text{if } \mathcal{C}[\![c_1]\!]\sigma = \perp \text{ then } \perp \text{ else } \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \\
\mathcal{C}[\![\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2]\!] &= \lambda\,\sigma\,.\,\text{if } \mathcal{B}[\![b]\!]\sigma = 0 \text{ then } \mathcal{C}[\![c_2]\!]\sigma \text{ else } \mathcal{C}[\![c_1]\!]\sigma \\
\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] &= \mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))
\end{aligned}$$

$$\Gamma \in (\Sigma \to \mathbb{B}) \times (\Sigma \to \Sigma_\perp) \to [(\Sigma \to \Sigma_\perp) \to (\Sigma \to \Sigma_\perp)]$$

$$\Gamma(\beta, \phi) = \lambda\,\psi\,.\,\lambda\,\sigma\,.\,\text{if } \beta\sigma = 0 \text{ then } \sigma \text{ else if } \phi\sigma = \perp \text{ then } \perp \text{ else } \psi(\phi\sigma)$$

Abbildung 6.2: Denotationale Semantik von IMP

Implementierung in Standard ML

Die Syntax und die denotationale Semantik von IMP lassen sich direkt in Standard ML implementieren. Dabei kann man den für Schleifen benötigten Fixpunktoperator durch die Prozedur `fix` aus Abschnitt 5.2 realisieren. Sie sollten diese Übungsaufgabe unbedingt vollständig ausführen. Sie lernen dabei, dass es sich bei unseren Definitionen für die Syntax und Semantik von IMP um funktionale Programme in mathematischer Notation handelt. Wenn wir diese Definitionen in Standard ML schreiben, können wir ihre Typkonsistenz automatisch überprüfen lassen. Zudem können wir die definierten Prozeduren ausführen. Die Prozedur für die Denotationsfunktion $\mathcal{C}$ liefert uns einen Interpreter für Kommandos. Wir können uns also durch Experimente von der computationalen Korrektheit der Semantik überzeugen.

Die Implementierung der denotationalen Semantik in Standard ML unterscheidet sich gegenüber der mathematischen Formulierung dadurch, dass die Prozedur, die die Denotation eines Kommandos berechnet, divergiert, falls das Kommando divergiert. Es stellt sich die Frage, ob man die denotationale Semantik so implementieren kann, dass die für ein Kommando berechnete Prozedur immer terminiert (indem sie für divergierende Kommandos einen Wert liefert, der $\perp$ entspricht). Wir werden später sehen, dass dies unmöglich ist, da das Halteproblem von IMP unentscheidbar ist.

Zwei Propositionen

Die folgende Propositionen formulieren zwei wichtige Eigenschaften der Denotation von Schleifen.

Proposition 6.1.1 *Seien $b \in Bexp$, $c \in Com$ und $\sigma, \sigma' \in \Sigma$. Dann:*

$$\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma = \sigma' \iff \exists n \in \mathbb{N}\colon\ (\Gamma(\mathcal{B}[\![b]\!], \mathcal{C}[\![c]\!]))^n\,(\lambda\,\sigma\,.\,\perp)\,\sigma = \sigma'$$

Beweis Folgt sofort aus der Definition von $\mathcal{C}$ und den Eigenschaften der VPO $\Sigma \to \Sigma_\perp$. □

Proposition 6.1.2 *Sei $b \in Bexp$ und $c \in Com$. Dann:*

$$\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] = \mathcal{C}[\![\texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip}]\!]$$

Beweis Sei $\sigma \in \Sigma$. Dann:

$$\begin{aligned}
\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma &= \mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ \sigma \\
&= (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ (\mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!])))\ \sigma \\
&= (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ (\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!])\ \sigma \\
&= \text{if } \mathcal{B}[\![b]\!]\sigma = 0 \text{ then } \sigma \\
&\quad\ \text{else if } \mathcal{C}[\![c]\!]\sigma = \bot \text{ then } \bot \\
&\qquad\quad \text{else } (\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!])(\mathcal{C}[\![c]\!]\sigma) \\
&= \text{if } \mathcal{B}[\![b]\!]\sigma = 0 \text{ then } \mathcal{C}[\![\texttt{skip}]\!]\sigma \\
&\quad\ \text{else } \mathcal{C}[\![c;\ \texttt{while}\ b\ \texttt{do}\ c]\!]\sigma \\
&= \mathcal{C}[\![\texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip}]\!]\sigma
\end{aligned}$$

Die zweite Gleichung folgt aus der Fixpunkteigenschaft von *fix*, alle anderen Gleichungen folgen direkt aus den Definitionen. □

Äquivalenz und Ersetzungseigenschaften

Wir haben die denotationale Semantik von IMP nach einem Muster definiert, dass wir bereits mehrfach angewendet haben (zuletzt für die aussagenlogische Sprache AL). Dieses Muster gibt uns auch entsprechende Definitionen für Äquivalenz und Substitution vor und garantiert die einschlägigen Ersetzungseigenschaften.

Da wir diesmal drei syntaktische Klassen mit getrennten Denotationsfunktionen haben, bekommen wir auch drei Äquivalenzrelationen:

$$\begin{aligned}
a_1 \models\!\!\!| \ a_2 &\iff \mathcal{A}[\![a_1]\!] = \mathcal{A}[\![a_2]\!] \\
b_1 \models\!\!\!| \ b_2 &\iff \mathcal{B}[\![b_1]\!] = \mathcal{B}[\![b_2]\!] \\
c_1 \models\!\!\!| \ c_2 &\iff \mathcal{C}[\![c_1]\!] = \mathcal{C}[\![c_2]\!]
\end{aligned}$$

Variablen gibt es nur bei arithmetischen Ausdrücken, sie fehlen bei Booleschen Ausdrücken und Kommandos. Es gelten die folgenden Substitutionseigenschaften:

$$\begin{aligned}
\mathcal{A}[\![a[a'/X]]\!]\sigma &= \mathcal{A}[\![a]\!](\sigma[\mathcal{A}[\![a']\!]\sigma/X]) \\
\mathcal{B}[\![b[a'/X]]\!]\sigma &= \mathcal{B}[\![b]\!](\sigma[\mathcal{A}[\![a']\!]\sigma/X]) \\
\mathcal{C}[\![c[a'/X]]\!]\sigma &= \mathcal{C}[\![c]\!](\sigma[\mathcal{A}[\![a']\!]\sigma/X])
\end{aligned}$$

6.2 Operationale Semantik

Wir definieren jetzt eine zweite Semantik für IMP, die als operationale Sematik bezeichnet wird. Dazu definieren wir mithilfe von Inferenzregeln eine Relation

$$OCom \subseteq \Sigma \times Com \times \Sigma$$

so dass

$$\langle\sigma, c, \sigma'\rangle \in OCom \iff \mathcal{C}[\![c]\!]\sigma = \sigma'$$

gilt. Statt $\langle\sigma, c, \sigma'\rangle \in OCom$ werden wir

$$\sigma \vdash c \Rightarrow \sigma'$$

schreiben. Die Aussage $\sigma \vdash c \Rightarrow \sigma'$ soll also genau dann gelten, wenn die Ausführung des Kommandos c für den Anfangszustand σ mit dem Endzustand σ' terminiert.

Die operationale Sematik von IMP wird durch die Inferenzregeln in Abbildung 6.3 definiert. Dabei verwenden wir die bereits definierten Denotationsfunktionen für arithmetische und Boolesche Ausdrücke. Selbstverständlich kann man die Semantik dieser Ausdrücke ebefalls operational definieren (siehe [Winskel]). Dabei bekommt man zwei Relationen

$$\begin{aligned} OAexp &\subseteq \Sigma \times Aexp \times \mathbb{Z} \\ OBexp &\subseteq \Sigma \times Bexp \times \mathbb{B} \end{aligned}$$

Im nächsten Abschnitt werden wir beweisen, dass die denotationale und die operationale Semantik von Kommandos übereinstimmen. Übereinstimmung bedeutet dabei, dass für alle $c \in Com$ und alle $\sigma, \sigma' \in \Sigma$ gilt:

$$\sigma \vdash c \Rightarrow \sigma' \iff \mathcal{C}[\![c]\!]\sigma = \sigma'$$

Wir gehen jetzt auf die Unterschiede zwischen den beiden Sematiken ein:

1. Die denotationale Semantik arbeitet mit funktionaler Notation, die operationale Semantik mit relationaler Notation.
2. Die denotationale Semantik verwendet zwei getrennte Rekursionen: Strukturelle Rekursion für die Syntax und Fixpunkt-Rekursion für die Denotation von Schleifen. Die operationale Semantik behandelt diese zwei Aspekte mit nur einer Rekursion.

$$\frac{\mathcal{A}[\![a]\!]\sigma = n \qquad \sigma' = \sigma[n/X]}{\sigma \vdash X := a \Rightarrow \sigma'}$$

$$\frac{\sigma \vdash c_1 \Rightarrow \sigma' \qquad \sigma' \vdash c_2 \Rightarrow \sigma''}{\sigma \vdash c_1; c_2 \Rightarrow \sigma''}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = 0 \qquad \sigma \vdash c_2 \Rightarrow \sigma'}{\sigma \vdash \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \Rightarrow \sigma'}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = 1 \qquad \sigma \vdash c_1 \Rightarrow \sigma'}{\sigma \vdash \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \Rightarrow \sigma'}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = 0}{\sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = 1 \qquad \sigma \vdash c \Rightarrow \sigma' \qquad \sigma' \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma''}{\sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma''}$$

Abbildung 6.3: Operationale Semantik von Kommandos

3. Die denotationale Semantik ist wegen der in (1) und (2) dargelegten Eigenschaften eine bessere Grundlage für Beweise.

Mathematisch gesehen ist die denotationale Semantik von IMP der operationalen Semantik von IMP klar überlegen. Bei komplexeren Programmiersprachen bekommen wir jedoch ein anderes Bild, da sich ihre Semantik of nur schwer oder gar nicht denotational beschreiben lässt. Dagegen ist die operationale Methode auch für komplexere Programmiersprachen anwendbar.

Die Semantik von Standard ML ist operational beschrieben. Die Semantik von nebenläufigen Sprachen entzieht sich in der Regel einer denotationalen Beschreibung.

6.3 Übereinstimmung der Semantiken

Wir führen den Übereinstimmungsbeweis in zwei Teilen.

Proposition 6.3.1 $\forall \sigma \in \Sigma\ \forall c \in \mathit{Com}\ \forall \sigma' \in \Sigma\colon\ \sigma \vdash c \Rightarrow \sigma' \Rightarrow \mathcal{C}[\![c]\!]\sigma = \sigma'$

Beweis Durch Regelinduktion über die Definition der operationalen Semantik. Von besonderem Interesse sind die Regeln für Schleifen, die Beweisteile für die anderen Regeln sind einfach. Wir beschränken uns auf die Beweisteile für die Regeln für Sequentialisierungen und Schleifen.

1. *Regel für Sequentialisierung.* Wir nehmen an:

   $$\mathcal{C}[\![c_1]\!]\sigma = \sigma' \ \wedge\ \ \mathcal{C}[\![c_2]\!]\sigma' = \sigma''$$

   Wir müssen zeigen:

   $$\mathcal{C}[\![c_1; c_2]\!]\sigma = \sigma''$$

   Das folgt mit der Definition von $\mathcal{C}$ und den Annahmen:

   $$\begin{aligned} \mathcal{C}[\![c_1; c_2]\!]\sigma &= \text{if } \mathcal{C}[\![c_1]\!]\sigma = \bot \text{ then } \bot \text{ else } \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \\ &= \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \\ &= \sigma'' \end{aligned}$$

2. *Erste Regel für Schleifen.* Wir nehmen an:

   $$\mathcal{B}[\![b]\!]\sigma = 0$$

Wir müssen zeigen:

$$\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma = \sigma$$

Das tun wir unter Benutzung der Fixpunkteigenschaft wie folgt:

$$\begin{aligned}
\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma &= \mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ \sigma \\
&= (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ (\mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!])))\ \sigma \\
&= \text{if } \mathcal{B}[\![b]\!]\sigma = 0 \text{ then } \sigma \text{ else } \cdots \\
&= \sigma
\end{aligned}$$

3. *Zweite Regel für Schleifen.* Wir nehmen an:

$$\mathcal{B}[\![b]\!]\sigma = 1\ \wedge\ \mathcal{C}[\![c]\!]\sigma = \sigma'\ \wedge\ \mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma' = \sigma''$$

Wir müssen zeigen:

$$\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma = \sigma''$$

Das tun wir unter Benutzung der Fixpunkteigenschaft wie folgt:

$$\begin{aligned}
\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma &= \mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ \sigma \\
&= (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ (\mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!])))\ \sigma \\
&= \text{if } \mathcal{B}[\![b]\!]\sigma = 0 \text{ then } \sigma \\
&\qquad \text{else if } \mathcal{C}[\![c]\!]\sigma = \bot \text{ then } \bot \\
&\qquad\qquad \text{else } (\mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!])))\ (\mathcal{C}[\![c]\!]\sigma) \\
&= \mathit{fix}\ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ \sigma' \\
&= \mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma' \\
&= \sigma''
\end{aligned}$$

□

Proposition 6.3.2 $\forall\sigma \in \Sigma\ \forall c \in \mathit{Com}\ \forall\sigma' \in \Sigma\colon\ \mathcal{C}[\![c]\!]\sigma = \sigma' \Rightarrow \sigma \vdash c \Rightarrow \sigma'$

Beweis Durch strukturelle Induktion über *Com*. Wer benötigen also einen Beweisteil für jede Gleichung der denotationalen Semantik. Interessant ist der Beweisteil für Schleifen, die restlichen Beweisteile sind einfach. Wir zeigen die Beweisteile für Sequentialsierung und Schleifen.

1. Sei $\mathcal{C}[\![c_1; c_2]\!]\sigma = \sigma'$. Es genügt zu zeigen, dass ein $\sigma'' \in \Sigma$ existiert mit

$$\sigma \vdash c_1 \Rightarrow \sigma''\ \wedge\ \sigma'' \vdash c_2 \Rightarrow \sigma'$$

Mit der Annahme und der Definition von $\mathcal{C}$ folgt:

$$\begin{aligned}\sigma' &= \mathcal{C}[\![c_1; c_2]\!]\sigma \\ &= \text{if } \mathcal{C}[\![c_1]\!]\sigma = \bot \text{ then } \bot \text{ else } \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma) \\ &= \mathcal{C}[\![c_2]\!](\mathcal{C}[\![c_1]\!]\sigma)\end{aligned}$$

Also existiert ein σ'' mit

$$\mathcal{C}[\![c_1]\!]\sigma = \sigma'' \ \wedge\ \mathcal{C}[\![c_2]\!]\sigma'' = \sigma'$$

Jetzt liefert die Induktionsannahme

$$\sigma \vdash c_1 \Rightarrow \sigma'' \ \wedge\ \sigma'' \vdash c_2 \Rightarrow \sigma'$$

2. Sei $\mathcal{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\sigma = \sigma'$. Wir müssen zeigen, dass

$$\sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma'$$

gilt. Wegen Proposition 6.1.1 genügt es, die folgende Aussage zu zeigen:

$$\begin{aligned}&\forall n \in \mathbb{N}\ \forall \sigma \in \Sigma\ \forall \sigma' \in \Sigma : \\ &\quad \sigma' = (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))^n(\lambda\,\sigma\,.\,\bot))\ \sigma \ \Rightarrow\ \sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma'\end{aligned}$$

Diese Behauptung zeigen wir durch Induktion über $n \in \mathbb{N}$.

Sei $n = 0$. Dann ist die Implikation trivialerweise erfüllt, da $\sigma' \neq \bot$.

Sei $n > 0$. Weiter sei $\sigma' = (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))^n(\lambda\,\sigma\,.\,\bot)\ \sigma$. Dann

$$\sigma' \ = \ (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))\ ((\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))^{n-1}(\lambda\,\sigma\,.\,\bot))\ \sigma$$

Wir unterscheiden jetzt zwei Fälle:

a) *Sei* $\mathcal{B}[\![b]\!]\sigma = 0$. Dann $\sigma = \sigma'$ nach Definition von Γ. Also folgt $\sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma'$ mit der ersten Regel für Schleifen.

b) *Sei* $\mathcal{B}[\![b]\!]\sigma = 1$. Dann folgt aus der Definition von Γ, dass ein σ'' existiert mit

$$\sigma'' = \mathcal{C}[\![c]\!]\sigma$$

und

$$\sigma' = (\Gamma(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]))^{n-1}(\lambda\,\sigma\,.\,\bot)\ \sigma''$$

Mit der äußeren Induktionsannahme folgt

$$\sigma \vdash c \Rightarrow \sigma''$$

Mit der inneren Induktionsannahme folgt

$$\sigma'' \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma'$$

Also haben wir $\sigma \vdash \texttt{while}\ b\ \texttt{do}\ c \Rightarrow \sigma'$ mit der zweiten Regel für Schleifen.

□

Satz 6.3.3 (Übereinstimmung) *Für alle $c \in Com$ und alle $\sigma, \sigma' \in \Sigma$ gilt:*

$$\sigma \vdash c \Rightarrow \sigma' \iff \mathcal{C}[\![c]\!]\sigma = \sigma'$$

Beweis Proposition 6.3.1 und Proposition 6.3.2. □

Aus der Übereinstimmung der Semantiken bekommen wir sofort eine wichtige Eigenschaft der operationalen Semantik.

Korollar 6.3.4 *Für jedes Kommando $c \in Com$ und jeden Zustand $\sigma \in \Sigma$ gibt es höchstes einen Zustand $\sigma' \in \Sigma$ mit $\sigma \vdash c \Rightarrow \sigma'$.*

