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 11: Kontexte Besprechung: 03.07.2012

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

(a) `(if (true) then x := 18 else []); (y := 42; [])` ist ein Kontext.
(b) Wenn $\langle c,\ \sigma\rangle \overset{\infty}{\to}_1$, dann $\mathcal{D}[\![c]\!]\,\sigma = \bot$.
(c) Wenn $K_1[c] = K_2[c]$, dann $K_1 = K_2$.
(d) Wenn $\langle c,\ \sigma\rangle \to_1 \langle c',\ \sigma'\rangle$, dann $\langle K[c],\ \sigma\rangle \to_1 \langle K[c'],\ \sigma'\rangle$.
(e) $\mathcal{D}[\![\_]\!]$ ist surjektiv.

Lösung:

(1a) Falsch. Ein Kontext kann per Definition nur ein Loch enthalten.

(1b) Richtig. Wenn $\langle c,\ \sigma\rangle \overset{\infty}{\to}_1$, dann gibt es kein σ' mit $\langle c,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$, also nach Thm. 37 auch kein σ' mit $\mathcal{D}[\![c]\!]\,\sigma = \sigma'$. Damit bleibt also nur $\mathcal{D}[\![c]\!]\,\sigma = \bot$.

(1c) Falsch. Gegenbeispiel: $K_1 = [];\ c,\ K_2 = c;\ []$, wobei c beliebig.
Die Kontextfüllfuntion $\_[\_]$ ist also nicht injektiv im ersten Argument, aber im zweiten: Wenn $K[c_1] = K[c_2]$, dann $c_1 = c_2$. Beweis: Induktion über K.
Außerdem gilt: Wenn $K_1[c] = K_2[c]$ für alle c, dann gilt $K_1 = K_2$.
Beweis: Induktion über K_1 (K_2 beliebig), dann jeweils Fallunterscheidung über K_2 und geeignete Wahl von c für die unmöglichen Fälle.

(1d) Falsch. Beispiel: $K = \texttt{if (true) then [] else skip}$, $c = \texttt{x := 1}$.
Dann $\langle c,\ \sigma\rangle \to_1 \langle \texttt{skip},\ \sigma[\texttt{x} \mapsto 1]\rangle$, aber nur $\langle K[c],\ \sigma\rangle \to_1 \langle \texttt{x := 1},\ \sigma\rangle$.

(1e) Falsch. Nicht jede Funktion ist berechenbar, jedoch sind alle durch While-Programme ausdrückbaren Funktionen berechenbar. Dies kann man mit einem einfachen Kardinalitätsargument beweisen, oder auch mit der Unentscheidbarkeit des Halteproblems. Sind die semantischen Objekte so definiert dass es keine gibt, die nicht schon syntaktisch Sinn machen, so spricht man von *Universalität*.

2. Programmäquivalenz (H)

Verschiedene Semantiken führen zu verschiedenen Äquivalenzbegriffen für Programme. c_1 und c_2 heißen *äquivalent bzgl. der*

Big-Step-Semantik wenn für alle σ, σ' gilt, dass $\langle c_1,\ \sigma\rangle \Downarrow \sigma'$ gdw. $\langle c_2,\ \sigma\rangle \Downarrow \sigma'$;

Small-Step-Semantik wenn für alle σ und σ' gilt,
dass $\langle c_1,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ gdw. $\langle c_2,\ \sigma\rangle \overset{*}{\to}_1 \langle \texttt{skip},\ \sigma'\rangle$ und dass $\langle c_1,\ \sigma\rangle \overset{\infty}{\to}_1$ gdw. $\langle c_2,\ \sigma\rangle \overset{\infty}{\to}_1$;

denotationalen Semantik wenn $\mathcal{D}[\![c_1]\!] = \mathcal{D}[\![c_2]\!]$.

c_1 und c_2 heißen *beobachtungsgleich* bezüglich einer Menge M von Kontexten, falls $K[c_1]$ und $K[c_2]$ semantisch äquivalent sind für alle Kontexte $K \in M$.

(a) Vergleichen Sie die verschiedenen Äquivalenzbegriffe der Semantiken.

(b) Finden Sie eine (möglichst kleine) Menge von Kontexten, für die beobachtungsgleiche Programme auch semantisch äquivalent sind.

(c) Für welche M sind äquivalente Programme beobachtungsgleich?

(d) Zeigen Sie, dass folgende Programme äquivalent sind:

```
while (not x == 1) do if (odd(x)) then x := (x * 3) + 1 else x := x div 2
```

und

```
while (not x == 1) do if (not odd(x)) then x := x div 2 else x := (x * 3) + 1
```

wobei a_1 `div` a_2 bzw. `odd(`a`)` ein neuer arithmetischer bzw. boolescher Ausdruck mit folgender Semantik sei:

$$\mathcal{A}[\![a_1 \texttt{ div } a_2]\!]\sigma = \begin{cases} 0 & \text{falls } \mathcal{A}[\![a_2]\!]\sigma = 0 \\ \left\lfloor \frac{\mathcal{A}[\![a_1]\!]\sigma}{\mathcal{A}[\![a_2]\!]\sigma} \right\rfloor & \text{sonst} \end{cases}$$

$$\mathcal{B}[\![\texttt{odd(}a\texttt{)}]\!]\sigma = (\mathcal{A}[\![a]\!]\sigma \text{ ungerade})$$

Lösung:

(2a) In einer früheren Aufgabe haben wir schon gesehen, dass die semantische Äquivalenz bei Big-Step- und Small-Step-Semantik für unsere einfache While-Sprache gleich ist. Thm. 37 zeigt, dass auch die denotationale Semantik für diese Sprache den gleichen Äquivalenzbegriff liefert.

(2b) Wähle $M = \{\, [\,] \,\}$.

(2c) Dies gilt für alle M – das ist der Kern der Kompositionalität.
Beweis: Sei $K \in M$ und $\mathcal{D}[\![c_1]\!] = \mathcal{D}[\![c_2]\!]$. Dann folgt mit Thm. 38:

$$\mathcal{D}[\![K\,[c_1]]\!] = \mathcal{K}[\![K]\!]\,\mathcal{D}[\![c_1]\!] = \mathcal{K}[\![K]\!]\,\mathcal{D}[\![c_2]\!] = \mathcal{D}[\![K\,[c_2]]\!]$$

(2d) Am einfachsten ist der Beweis über die denotationale Semantik mit dem Kontext

$$K = \texttt{while (not x == 1) do } [\,]$$

Dann muss man nur noch die Äquivalenz des Schleifenrumpfes zeigen, was aber einfach durch Ausrechnen geht.
Hätte man keine denotationale, kompositionale Semantik zur Verfügung, wäre der Beweis wesentlich aufwändiger. Bei der Big-Step-Semantik bräuchte man zwei Regel-Induktionen über die Big-Step-Semantik, bei der Small-Step-Semantik müsste man eine Bisimulationsrelation finden, die unter den einzelnen Schritten abgeschlossen ist.

Anmerkung: Üblicherweise ist Beobachtungsäquivalenz und semantische Äquivalenz nicht das gleiche. In der Regel verlangt man für Beobachtungsäquivalenz, dass die erhaltenen Programme geschlossen sind, d.h. weder von Eingaben abhängen und nur endlich viel Information ausgeben. Bei unserer While-Sprache sind solche Einschränkungen aber nicht einfach zu realisieren, mit den Exceptions und Continuations sähe das schon wieder anders aus. Bei funktionalen Sprachen nimmt man als Kontexte nur solche, die zu einem Wert auswerten - dadurch kann man ggf. zu anderen Äquivalenzbegriffen als semantischer Äquivalenz kommen.

3. Nichtdeterminismus (Ü)

Die Sprache While_{ND} erweitert While um den nichtdeterministischen Auswahloperator c_1 `or` c_2. In der Vorlesung haben wir bereits die Big-Step- und Small-Step-Semantiken darauf erweitert, hier sollen Sie nun eine denotationale Erweiterung entwickeln.

Statt höchstens eines Endzustandes sind nun beliebig viele Endzustände möglich. Die Bedeutung eines Programms ändert sich dementsprechend von einer partiellen Funktion auf Zuständen ($\Sigma \rightharpoonup \Sigma$) zu einer Funktion, die einen Anfangszustand auf eine Menge möglicher Endzustände ($\Sigma \Rightarrow \mathfrak{P}(\Sigma)$) abbildet.

(a) Passen Sie die Definition von $\mathcal{D}[\![\_]\!]$ entsprechend an. Definieren Sie auch $\mathcal{D}[\![c_1 \texttt{ or } c_2]\!]$.

(b) Für den Fixpunkt benötigen Sie auch eine neue Approximationsordnung $\sqsubseteq$:

$$f \sqsubseteq g \quad \text{gdw.} \quad \forall \sigma.\ f(\sigma) \subseteq g(\sigma)$$

Was müssen Sie neu zeigen, um die Existenz und Eindeutigkeit des Fixpunkts zu garantieren?

(c) Berechnen Sie $\mathcal{D}[\![\texttt{skip or while (true) do skip}]\!]$.

(d) Reformulieren Sie das Adäquatheitstheorem. Gilt es immer noch?

(e) Wiederholen Sie Aufgabe 2a für die Semantiken von While_{ND}.

(f) Definieren Sie eine Funktion $\mathcal{T}[\![\_]\!] :: \mathsf{Com} \Rightarrow \mathfrak{P}(\Sigma)$, die die Menge der Zustände berechnet, für die die übergebene Anweisung immer terminiert.

Lösung:

(3a)

$$\begin{aligned}
\mathcal{D}[\![\texttt{skip}]\!] &= \lambda\sigma.\ \{\,\sigma\,\} \\
\mathcal{D}[\![x \texttt{ := } a]\!] &= \lambda\sigma.\ \{\,\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]\,\} \\
\mathcal{D}[\![c_1\texttt{; } c_2]\!] &= \mathcal{D}[\![c_1]\!] \circ \mathcal{D}[\![c_2]\!] \\
\mathcal{D}[\![\texttt{if (}b\texttt{) then } c_1 \texttt{ else } c_2]\!] &= \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ \mathcal{D}[\![c_1]\!],\ \mathcal{D}[\![c_2]\!]) \\
\mathcal{D}[\![\texttt{while (}b\texttt{) do } c]\!] &= \mathrm{FIX}\,(\lambda f.\ \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ f \circ \mathcal{D}[\![c]\!],\ \lambda\sigma.\{\,\sigma\,\})) \\
\mathcal{D}[\![c_1 \texttt{ or } c_2]\!] &= \mathcal{D}[\![c_1]\!] \sqcup \mathcal{D}[\![c_2]\!]
\end{aligned}$$

wobei

$$(f \circ g)(\sigma) = \bigcup\{\, f(\sigma') \mid \sigma' \in g(\sigma)\,\} = \bigcup_{\sigma' \in g(\sigma)} f(\sigma')$$

$$\mathrm{IF}\,(p,\ f,\ g)\,\sigma = \begin{cases} f(\sigma) & \text{falls } p(\sigma) = \mathbf{tt} \\ g(\sigma) & \text{falls } p(\sigma) = \mathbf{ff} \end{cases}$$

$$(f \sqcup g)(\sigma) = f(\sigma) \cup g(\sigma)$$

(3b) Man muss die Anwendbarkeit des Knaster-Tarski-Fixpunktsatzes (Thm. 33) zeigen:

- $(\Sigma \Rightarrow \mathfrak{P}(\Sigma), \sqsubseteq)$ ist eine ccpo
- $F(f) = \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ f \circ \mathcal{D}[\![c]\!],\ \lambda\sigma.\ \{\,\sigma\,\})$ ist monoton und kettenstetig.

Lemma: $(\Sigma \Rightarrow \mathfrak{P}(\Sigma), \sqsubseteq)$ ist eine ccpo.
Beweis. $(\mathfrak{P}(\Sigma), \subseteq)$ ist eine ccpo mit $\bigsqcup Y = \bigcup Y$. $\sqsubseteq$ überträgt die ccpo $\subseteq$ punktweise auf den Funktionenraum. Das ist dann immer auch eine ccpo mit $(\bigsqcup Y)\,(\sigma) = \bigsqcup\{\, f(\sigma) \mid f \in Y\,\}$, wie wir in der Vorlesung gesehen haben (Lemma 41). □
Lemma: F ist monoton und kettenstetig. Der Beweis verläuft ganz analog zu Thm. 34.

(3c)

$$\begin{aligned}
&\mathcal{D}[\![\texttt{skip or while (true) do skip}]\!]\,(\sigma) \\
&= (\mathcal{D}[\![\texttt{skip}]\!] \sqcup \mathcal{D}[\![\texttt{while (true) do skip}]\!])(\sigma) \\
&= \mathcal{D}[\![\texttt{skip}]\!]\,\sigma \cup \mathcal{D}[\![\texttt{while (true) do skip}]\!]\,\sigma \\
&= \{\,\sigma\,\} \cup \mathrm{FIX}\,(\lambda f.\ \mathrm{IF}\,(\mathcal{B}[\![\texttt{true}]\!],\ f \circ \mathcal{D}[\![\texttt{skip}]\!],\ \lambda\sigma.\ \{\,\sigma\,\})) \\
&= \{\,\sigma\,\} \cup \mathrm{FIX}\,(\lambda f.\ \mathrm{IF}\,(\lambda\sigma.\ \mathbf{tt},\ f \circ (\lambda\sigma.\ \{\,\sigma\,\}),\ \lambda\sigma.\ \{\,\sigma\,\})) \\
&= \{\,\sigma\,\} \cup \mathrm{FIX}\,(\lambda f.\ f) = \{\,\sigma\,\} \cup \bot = \{\,\sigma\,\}
\end{aligned}$$

(3d) Wenn $\langle c,\ \sigma\rangle \Downarrow \sigma'$ oder $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma'\rangle$, dann $\sigma' \in \mathcal{D}\,[\![c]\!]\,\sigma$.
Wenn $\sigma' \in \mathcal{D}\,[\![c]\!]\,\sigma$, dann $\langle c,\ \sigma\rangle \Downarrow \sigma'$ und $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma'\rangle$.

Auch in While_{ND} gilt, dass $\langle c,\ \sigma\rangle \Downarrow \sigma'$ gdw. $\langle c,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{skip},\ \sigma'\rangle$ (vgl. Vorlesung). Damit muss man wie in der Vorlesung nur den Beweis bezüglich der Big-Step-Semantik neu durchführen: Induktion über die Big-Step-Semantik bzw. Induktion über c. Die Beweise verlaufen entsprechend!

(3e) Aus dem Adäquatheitstheorem von 3d folgt, dass semantische Äquivalenz bei denotationaler Semantik und bei Big-Step-Semantik das gleiche ist. Semantische Äquivalenz bgzl. der Small-Step-Semantik ist feiner als semantische Äquivalenz bezüglich der anderen Semantiken, da sie auch potenzielle Nichttermination berücksichtigt.

Beispiel: $c_1 = \texttt{skip}$, $c_2 = \texttt{skip or while (true) do skip}$. Es gilt $\mathcal{D}\,[\![c_1]\!]\,\sigma = \{\,\sigma\,\}$ und $\mathcal{D}\,[\![c_2]\!]\,\sigma = \{\,\sigma\,\}$ für alle σ. Somit $\mathcal{D}\,[\![c_1]\!] = \mathcal{D}\,[\![c_2]\!]$, aber $\langle c_2,\ \sigma\rangle \xrightarrow{\infty}_1$ und $\langle c_1,\ \sigma\rangle \not\xrightarrow{\infty}_1$.

(3f)

$$
\begin{aligned}
\mathcal{T}[\![\texttt{skip}]\!] &= \Sigma \\
\mathcal{T}[\![x\ \texttt{:=}\ a]\!] &= \Sigma \\
\mathcal{T}[\![c_1\texttt{;}\ c_2]\!] &= \{\,\sigma \in \mathcal{T}[\![c_1]\!] \mid \mathcal{D}\,[\![c_1]\!]\,\sigma \subseteq \mathcal{T}[\![c_2]\!]\,\} \\
\mathcal{T}[\![\texttt{if (}b\texttt{) then }c_1\texttt{ else }c_2]\!] &= \{\sigma \mid (\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \implies \sigma \in \mathcal{T}[\![c_1]\!]) \wedge (\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff} \implies \sigma \in \mathcal{T}[\![c_2]\!])\,\} \\
\mathcal{T}[\![\texttt{while (}b\texttt{) do }c]\!] &= \mathrm{FIX}\,(\lambda M.\{\,\sigma \mid \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \implies \sigma \in \mathcal{T}[\![c]\!] \wedge \mathcal{D}\,[\![c]\!]\,\sigma \subseteq M\,\}) \\
\mathcal{T}[\![c_1\ \texttt{or}\ c_2]\!] &= \mathcal{T}[\![c_1]\!] \cap \mathcal{T}[\![c_2]\!]
\end{aligned}
$$

In der bisherigen Form konnte die denotationale Semantik nicht ausdrücken, dass ein Programm für einen Anfangszustand eventuell nicht terminiert. Mit der Funktion $\mathcal{T}[\![\_]\!]$ lässt sich dies aber ausdrücken. Damit kann man dann auch die denotationale Semantik eines Programms c zu einer Funktion $\Sigma \Rightarrow (\mathfrak{P}(\Sigma) \times \mathbb{B})$ erweitern, wobei das boolesche Flag angibt, ob c für den übergebenen Anfangszustand immer terminiert.

Für den Fixpunkt müsste man eigentlich wieder nachweisen, dass er existiert. Da $(\mathfrak{P}(\Sigma), \subseteq)$ eine ccpo ist, braucht man nur die Monotonie und Kettenstetigkeit des Funktionals

$$\Delta(M) = \{\,\sigma \mid \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \implies \sigma \in \mathcal{T}[\![c]\!] \wedge \mathcal{D}\,[\![c]\!]\,\sigma \subseteq M\,\}$$

nachweisen.

Die folgenden Beweise sind nur der Vollständigkeit halber und nicht für die Übung selbst. Δ ist nur kettenstetig, wenn der Schleifenrumpf endlich nichtdeterministisch ist, d.h., wenn $\sigma \in \mathcal{T}[\![c]\!]$, dann ist $\mathcal{D}\,[\![c]\!]\,\sigma$ endlich. Unter dieser Annahme lässt sich Kettenstetigkeit leicht zeigen:

- Monotonie: Sei $M \subseteq N$. Zu zeigen: $\Delta(M) \subseteq \Delta(N)$.
  Sei also σ beliebig mit $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \implies \sigma \in \mathcal{T}[\![c]\!] \wedge \mathcal{D}\,[\![c]\!] \subseteq M$. Dann gilt auch wegen $M \subseteq N$, dass $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \implies \sigma \in \mathcal{T}[\![c]\!] \wedge \mathcal{D}\,[\![c]\!] \subseteq N$, also $\sigma \in \Delta(N)$.
- Kettenstetigkeit:
  Kettenstetigkeit gilt nur, wenn immer terminierende Programme höchstens endlich viele mögliche Endzustände haben. Wir nehmen also an: Wenn $\sigma \in \mathcal{T}[\![c]\!]$, dann ist $\mathcal{D}\,[\![c]\!]\,\sigma$ endlich.
  Sei $Y \neq \emptyset$ eine nichtleere Kette. Zu zeigen: $\Delta\,(\bigcup Y) \subseteq \bigcup\{\,\Delta(M) \mid M \in Y\,\}$.
  Sei also $\sigma \in \Delta\,(\bigcup Y)$.
  - Fall $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff}$: Da Y nicht leer, gibt es ein $M \in Y$.
    Somit $\sigma \in \Delta(M)$, also $\sigma \in \bigcup\{\,\Delta(M) \mid M \in Y\,\}$.
  - Fall $\mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt}$: Dann gilt nach Voraussetzung, dass $\sigma \in \mathcal{T}[\![c]\!]$ und $\mathcal{D}\,[\![c]\!]\,\sigma \subseteq \bigcup Y$. Damit ist nach Annahme $\mathcal{D}\,[\![c]\!]\,\sigma$ endlich. Da Y eine nicht-leere Kette ist, gibt es damit ein $M \in Y$ mit $\mathcal{D}\,[\![c]\!]\,\sigma \subseteq M$. Zusammen mit $\sigma \in \mathcal{T}[\![c]\!]$ folgt, dass $\sigma \in \Delta(M)$. Damit folgt die Behauptung direkt.

Zum Beweis des endlichen Nichtdeterminismus ist allerdings die Infrastruktur aus der Vorlesung zu schwach: Man bräuchte dafür bereits eine Form von Fixpunkt-Induktion über Δ, die aber erst nach dem Nachweis der Kettenstetigkeit möglich ist. Allerdings ist Δ eine monotone Funktion auf $(\mathfrak{P}(\Sigma), \subseteq)$, was ein vollständiger Verband ist. Damit lässt sich der Fixpunkt $\mathrm{FIX}\,(\Delta)$ auch als $\bigcap\{\, M \mid \Delta(M) \subseteq M \,\}$ definieren (vgl. z.B. Winskel, Kap. 5.5), der nur die Monotonie von Δ benötigt.

Damit lässt sich dann auch der endliche Nichtdeterminismus durch Induktion über c zeigen:

- Fälle `skip` und x `:=` a: Trivial.
- Fall c_1`;` c_2:
  Induktionsannahmen: Für $i = 1, 2$ und alle σ gilt: Wenn $\sigma \in \mathcal{T}[\![c_i]\!]$, dann ist $\mathcal{D}[\![c_i]\!]\,\sigma$ endlich.
  Zu zeigen: Wenn $\sigma \in \mathcal{T}[\![c_1\texttt{;}\ c_2]\!]$, dann ist $\mathcal{D}[\![c_1\texttt{;}\ c_2]\!]$ endlich.
  Aus $\sigma \in \mathcal{T}[\![c_1\texttt{;}\ c_2]\!]$ folgt, dass $\sigma \in \mathcal{T}[\![c_1]\!]$ und $\mathcal{D}[\![c_1]\!]\,\sigma \subseteq \mathcal{T}[\![c_2]\!]$. Also ist $\mathcal{D}[\![c_1]\!]\,\sigma$ endlich. Nach Definition gilt $\mathcal{D}[\![c_1\texttt{;}\ c_2]\!]\,\sigma = \bigcup_{\sigma' \in \mathcal{D}[\![c_1]\!]\sigma} \mathcal{D}[\![c_2]\!]\,\sigma'$, dies ist also eine endliche Vereinigung. Es reicht also zu zeigen, dass $\mathcal{D}[\![c_2]\!]\,\sigma'$ für alle $\sigma' \in \mathcal{D}[\![c_1]\!]\,\sigma$ endlich ist. Dies folgt aber aus der Induktionsannahme, da $\mathcal{D}[\![c_1]\!]\,\sigma \subseteq \mathcal{T}[\![c_1]\!]$.
- Fall `if (`b`) then` c_1 `else` c_2:
  Induktionsannahmen: Für $i = 1, 2$ gilt: Wenn $\sigma \in \mathcal{T}[\![c_i]\!]$, dann ist $\mathcal{D}[\![c_i]\!]\,\sigma$ endlich.
  Zu zeigen: Wenn $\sigma \in \mathcal{T}[\![\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2]\!]$,
  dann ist $\mathcal{D}[\![\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2]\!]\,\sigma$ endlich.
  - Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt}$: Dann gilt nach Definition von $\mathcal{T}[\![\_]\!]$, dass $\sigma \in \mathcal{T}[\![c_1]\!]$. Nach Induktionsannahme ist damit $\mathcal{D}[\![c_1]\!]\,\sigma$ endlich. Mit $\mathcal{D}[\![\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2]\!]\,\sigma = \mathcal{D}[\![c_1]\!]\,\sigma$ folgt die Behauptung.
  - Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}$: Analog
- Fall `while (`b`) do` c:
  Induktionsannahme I: Für alle σ gilt: Wenn $\sigma \in \mathcal{T}[\![c]\!]$, dann ist $\mathcal{D}[\![c]\!]\,\sigma$ endlich.
  Zu zeigen: Wenn $\sigma \in \mathcal{T}[\![\texttt{while (}b\texttt{) do}\ c]\!] = \mathrm{FIX}\,(\Delta)$, dann ist $\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma$ endlich.
  Beweis durch Fixpunktindution über vollständigen Verbänden:
  Sei also σ beliebig mit $\sigma \in \Delta(\mathrm{FIX}\,(\Delta) \cap \{\, \sigma' \mid \mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma' \text{ endlich} \,\})$.
  Zu zeigen: $\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma$ ist endlich.
  - Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}$: Dann ist $\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma = \{\, \sigma \,\}$, also endlich.
  - Fall $\mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt}$:
    Nach Voraussetzung von σ ist also $\sigma \in \mathcal{T}[\![c]\!]$ und
    $$\mathcal{D}[\![c]\!]\,\sigma \subseteq \mathrm{FIX}\,(\Delta) \cap \{\, \sigma' \mid \mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma' \text{ endlich} \,\}$$
    Daraus ergibt sich die Induktionsannahme II:
    Für alle $\sigma' \in \mathcal{D}[\![c]\!]\,\sigma$ ist $\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma'$ endlich.
    Wegen $\sigma \in \mathcal{T}[\![c]\!]$ ist nach Induktionsannahme I $\mathcal{D}[\![c]\!]\,\sigma$ endlich. Wegen
    $$\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma = \bigcup_{\sigma' \in \mathcal{D}[\![c]\!]\sigma} \mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma'$$
    genügt es damit zu zeigen, dass $\mathcal{D}[\![\texttt{while (}b\texttt{) do}\ c]\!]\,\sigma'$ für alle $\sigma' \in \mathcal{D}[\![c]\!]\,\sigma$ endlich ist. Dies ist aber die Induktionsannahme II.
- Fall c_1 `or` c_2:
  Induktionsannahmen: Für $i = 1, 2$ gilt: Wenn $\sigma \in \mathcal{T}[\![c_i]\!]$, dann ist $\mathcal{D}[\![c_i]\!]\,\sigma$ endlich.
  Zu zeigen: Wenn $\sigma \in \mathcal{T}[\![c_1\ \texttt{or}\ c_2]\!]$, dann ist $\mathcal{D}[\![c_1\ \texttt{or}\ c_2]\!]\,\sigma$ endlich.
  Aus $\sigma \in \mathcal{T}[\![c_1\ \texttt{or}\ c_2]\!]$ folgt, dass $\sigma \in \mathcal{T}[\![c_1]\!]$ und $\sigma \in \mathcal{T}[\![c_2]\!]$. Damit sind $\mathcal{D}[\![c_1]\!]\,\sigma$ und $\mathcal{D}[\![c_2]\!]\,\sigma$ endlich. Also auch $\mathcal{D}[\![c_1\ \texttt{or}\ c_2]\!]\,\sigma = \mathcal{D}[\![c_1]\!]\,\sigma \cup \mathcal{D}[\![c_2]\!]\,\sigma$.

Den endlichen Nichtdeterminismus müssen wir hier nicht nur aus beweistechnischen Gründen fordern, sondern obige Definition für $\mathcal{T}$ wird falsch wenn wir unendlichen Nichtdeterminismus zulassen. Angenommen, wir haben einen Befehl x `:= randNat()` mit der Semantik

$\mathcal{D}[\![x \texttt{ := randNat())}]\!]\,\sigma = \{\sigma[x \mapsto n] \mid n \in \mathbb{N}\}$. Betrachte die Schleife c'

```
while (not x == 0) do (if (x <= -1) then x := randNat() else x := x - 1)
```

mit Bedingung b und Schleifenrumpf c. Es ist $\mathcal{T}[\![c]\!] = \Sigma$ und

$$\mathcal{D}[\![c]\!]\,\sigma = \begin{cases} \{\sigma[\mathtt{x} \mapsto n] \mid n \in \mathbb{N}\} & \text{falls } \sigma(\mathtt{x}) < 0 \\ \{\sigma[\mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]\} & \text{sonst.} \end{cases}$$

Beachte, dass $\mathcal{D}[\![c]\!]\,\sigma$ für $\sigma(\mathtt{x}) < 0$ stets unendlich ist und damit nie Teilmenge einer endlichen Menge. Damit erhalten wir folgendes Funktional

$$\Delta(M) = \{\sigma \mid \sigma(\mathtt{x}) = 0 \vee \mathcal{D}[\![c]\!]\,\sigma \subseteq M\}$$

mit den Iterationen (die offensichtlich alle endliche Mengen sind)

$$\begin{aligned} \Delta^0(\emptyset) &= \emptyset \\ \Delta^1(\emptyset) &= \{\sigma \mid \sigma(\mathtt{x}) = 0\} \\ \Delta^2(\emptyset) &= \{\sigma \mid \sigma(\mathtt{x}) = 0 \vee \sigma(\mathtt{x}) = 1\} \\ \Delta^3(\emptyset) &= \{\sigma \mid \sigma(\mathtt{x}) = 0 \vee \sigma(\mathtt{x}) = 1 \vee \sigma(\mathtt{x}) = 2\} \\ \Delta^i(\emptyset) &= \{\sigma \mid \sigma(\mathtt{x}) \in \{0, \ldots, i-1\}\}. \end{aligned}$$

Also ist $\mathcal{T}[\![c']\!] = \bigcup_{i \in \mathbb{N}} \Delta^i(\emptyset) = \{\sigma \mid \sigma(\mathtt{x}) \in \mathbb{N}\}$, obwohl die Schleife für alle Eingaben terminiert, also $\mathcal{T}[\![c']\!] = \Sigma$ gelten müsste. Beachte auch: Es gilt $\Delta(\mathcal{T}[\![c']\!]) = \Sigma$, das heißt $\bigcup_{i \in \mathbb{N}} \Delta^i(\emptyset)$ ist *kein* Fixpunkt von Δ.