

## Semantik von Programmiersprachen – SS 2017

<http://pp.ipd.kit.edu/lehre/SS2017/semantik>

### Lösungen zu Blatt 10: Fixpunkttheorie

Besprechung: 03.07.2017

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

- (a) Für  $f(\sigma) = \sigma[i \mapsto 1]$  und  $g(\sigma) = \sigma[i \mapsto 2]$  gilt  $f \sqsubseteq g$ .
- (b) Jede Teilmenge von  $\mathbb{R}_0^+ = \{x \in \mathbb{R} \mid x \geq 0\}$  hat bezüglich der normalen Ordnung  $\leq$  auf den reellen Zahlen  $\mathbb{R}$  ein kleinstes Element.
- (c) Jede Teilmenge einer total geordneten Menge ist eine Kette.
- (d) In einer ccpo  $(D, \sqsubseteq)$  hat jede Menge  $M \subseteq D$  eine obere Schranke.
- (e) Die Menge  $\mathfrak{P}^{fin}(\mathbb{N})$  der endlichen Teilmengen von  $\mathbb{N}$  ist mit der Teilmengenrelation  $\subseteq$  als Ordnung eine ccpo.
- (f) Jede ccpo  $(D, \sqsubseteq)$  hat ein kleinstes Element.
- (g) Das abgeschlossene Intervall  $[0, 1] \subseteq \mathbb{R}$  ist eine ccpo mit  $\leq$  als Ordnung.
- (h) IF  $(p, f, g)$  ist strikt in  $f$ .
- (i) Wenn  $f \circ g$  kettenstetig ist, dann sind auch  $f$  und  $g$  kettenstetig.

#### Lösung:

- (1a) Falsch. Die Approximationsordnung  $\sqsubseteq$  vergleicht nicht die Größe der Werte, die einer Variablen zugewiesen werden, sondern die Definiiertheit der Funktionen selbst.
- (1b) Falsch. Beispielsweise hat  $\{x \in \mathbb{R} \mid x > 0\}$  kein kleinstes Element, da 0 nicht enthalten ist. In  $\mathbb{R}_0^+$  hat zwar jede Menge  $M$  eine kleinste untere Schranke, aber diese muss nicht in  $M$  enthalten sein.  
 Nebenbemerkung: Nimmt man das Auswahlaxiom der Mengenlehre als gültig an, gibt es auf  $\mathbb{R}_0^+$  eine (von  $<$  verschiedene) Ordnung, bezüglich der jede Teilmenge ein kleinstes Element hat.
- (1c) Richtig. Eine Kette ist nichts anderes als eine total geordnete Menge. Teilmengen total geordneter Mengen bleiben total geordnet.
- (1d) Falsch. Sei  $D = \Sigma \rightarrow \Sigma$  und  $\sqsubseteq$  die Approximationsordnung. Dann gibt es keine obere Schranke für  $\{f, id\}$  mit  $f(\sigma) = \sigma[x \mapsto 1]$ .
- (1e) Falsch. Betrachte die Menge  $M = \{\{0, \dots, n\} \mid n \in \mathbb{N}\} \subseteq \mathfrak{P}^{fin}(\mathbb{N})$ . Dann ist  $M$  eine Kette bezüglich  $\subseteq$ . Die kleinste obere Schranke von  $M$  ist  $\mathbb{N}$ , aber  $\mathbb{N} \notin \mathfrak{P}^{fin}(\mathbb{N})$ .
- (1f) Richtig. Die leere Menge  $\emptyset$  ist immer eine Kette in  $(D, \sqsubseteq)$ .  $\bigsqcup \emptyset$  ist dann das kleinste Element von  $D$ :  
 Sei  $d \in D$  beliebig. Dann ist  $d$  eine obere Schranke von  $\emptyset$ . Da  $\bigsqcup \emptyset$  die kleinste obere Schranke von  $\emptyset$  ist, gilt  $\bigsqcup \emptyset \sqsubseteq d$ .
- (1g) Richtig.  $\mathbb{R}$  ist total geordnet, damit auch das Intervall, somit ist jede Teilmenge eine Kette. Sei also  $M \subseteq [0, 1]$ . Definiere

$$\bigsqcup M = \begin{cases} 0 & \text{falls } M = \emptyset \\ \sup(M) & \text{sonst} \end{cases}$$

wobei  $\sup(M)$  die kleinste obere Schranke in  $\mathbb{R}$  liefert (die für alle nicht-leeren, beschränkten Mengen  $M$  in  $\mathbb{R}$  existiert). Da  $M$  durch 0 und 1 beschränkt ist, muss  $0 \leq \sup(M) \leq 1$  gelten. Somit  $\bigsqcup M \in [0, 1]$ .

Anmerkung: Wenn das Intervall nicht abgeschlossen wäre, wäre es keine ccpo, weil entweder das kleinste Element (links offen) oder das größte Element (rechts offen) fehlt.

- (1h) Falsch. Das **if**-Konstrukt ist in allen (vernünftigen) Programmiersprachen nicht strikt – sonst müsste es immer sowohl **then**- als auch den **else**-Zweig auswerten. Thm. 109 zeigt zwar, dass  $\text{IF}(p, f, g)$  kettenstetig ist. Der Beweis funktioniert aber nicht, wenn man auch die leere Kette  $\emptyset$  für  $Y$  erlaubt, falls  $p \neq \lambda\sigma. \mathbf{tt}$  gilt. Hier ein Gegenbeispiel mit  $p(\sigma) = \mathbf{tt}$  und  $g = \text{id}$ .

$$\text{IF}(\lambda\sigma. \mathbf{ff}, \bigsqcup \emptyset, \text{id}) = \text{id} \not\sqsubseteq \perp = \bigsqcup \emptyset = \bigsqcup \{ \text{IF}(\lambda\sigma. \mathbf{ff}, f, \text{id}) \mid f \in \emptyset \}$$

- (1i) Falsch. Sei z.B.  $g = \perp$ ,  $f$  beliebig nicht kettenstetig. Dann ist  $f \circ g = \perp$  und  $\perp$  ist kettenstetig.

## 2. Monotonie und Fixpunkte (H)

Finden Sie eine Halbordnung  $(D, \sqsubseteq)$  mit kleinstem Element  $\perp$  und eine monotone Funktion  $f :: D \Rightarrow D$ , die mehrere Fixpunkte besitzt, aber keinen kleinsten.

**Lösung:** Das geforderte  $D$  kann nicht endlich sein, da alle endlichen Halbordnungen  $(D, \sqsubseteq)$  mit kleinstem Element kettenvollständig und alle monotonen Funktionen auf endlichen Halbordnungen  $(D, \sqsubseteq)$  automatisch kettenstetig sind.

Eine Lösung:

$$D = \mathbb{N} \cup \{A, B\}$$

$$m \sqsubseteq n = m \leq n, m \sqsubseteq A, m \sqsubseteq B, A \sqsubseteq A, B \sqsubseteq B \text{ für alle } n, m \in \mathbb{N}$$

$$f(n) = n + 1 \text{ für } n \in \mathbb{N}, f(A) = A, f(B) = B.$$

Dann hat  $f$  nur die Fixpunkte  $A$  und  $B$ , aber weder  $A \sqsubseteq B$  noch  $B \sqsubseteq A$ .

## 3. Exkurs: monadische Semantik (H)

In der funktionalen Programmierung kennt man das Abstraktionskonzept der *Monade*, die unter anderem durch die folgenden zwei Operationen charakterisiert werden kann:

$$\text{pure} :: a \Rightarrow m a$$

$$(>=>) :: (a \Rightarrow m b) \Rightarrow (b \Rightarrow m c) \Rightarrow (a \Rightarrow m c)$$

Statt auf  $\Sigma \rightarrow \Sigma$  können wir  $\mathcal{D}[\_]$  auf dem allgemeineren Typ  $\Sigma \Rightarrow m \Sigma$  definieren, angenommen  $m \Sigma$  ist eine ccpo<sup>1</sup>.

$$\mathcal{D}_m[\text{skip}] = \text{pure}$$

$$\mathcal{D}_m[x := a] = \text{pure} \circ \lambda\sigma. \sigma[x \mapsto \mathcal{A}[a] \sigma]$$

$$\mathcal{D}_m[c_1; c_2] = \mathcal{D}_m[c_1] >=> \mathcal{D}_m[c_2]$$

$$\mathcal{D}_m[\text{if } (b) \text{ then } c_1 \text{ else } c_2] = \text{IF}(\mathcal{B}[b], \mathcal{D}_m[c_1], \mathcal{D}_m[c_2])$$

$$\mathcal{D}_m[\text{while } (b) \text{ do } c] = \text{FIX}(\lambda f. \text{IF}(\mathcal{B}[b], \mathcal{D}_m[c] >=> f, \text{pure}))$$

$\mathcal{D}_{\text{Maybe}}[\_]$  entspricht damit der alten Semantik. Fallen Ihnen andere Monaden ein, die sinnvolle Semantiken ergeben?

<sup>1</sup>Man überlege sich, dass  $\Sigma \Rightarrow m \Sigma$  dann auch eine ccpo sein muss

**Lösung:** Ein paar Vorschläge:

Monade	Ordnung	modelliert
Maybe	Approximationsordnung	Nichttermination
Identity	Hat kein kleinstes Element!	
List	$\subseteq$	Nichtdeterminismus

Viele Erweiterungen lassen sich als Monad-Transformer modellieren und sind damit beliebig kombinierbar! Als Ordnung kann die der Basis-Monade *Maybe* benutzt werden.

Transformer	modelliert
ExceptT	Exceptions
ListT	Nichtdeterminismus
ConT	Continuations
ReaderT	Eingaben
WriterT	Ausgaben

#### 4. repeat c until b-Schleife (Ü)

In einer früheren Aufgabe haben wir schon die operationale Semantik einer **repeat**-Schleife betrachtet. Com wird dazu um das Syntaxkonstrukt **repeat c until b** erweitert und die operationale Big-Step-Semantik durch die Regeln

$$\text{REPEATTT: } \frac{\langle c, \sigma \rangle \Downarrow \sigma' \quad \mathcal{B} \llbracket b \rrbracket \sigma' = \mathbf{tt}}{\langle \text{repeat } c \text{ until } b, \sigma \rangle \Downarrow \sigma'}$$

$$\text{REPEATFF: } \frac{\langle c, \sigma \rangle \Downarrow \sigma' \quad \mathcal{B} \llbracket b \rrbracket \sigma' = \mathbf{ff} \quad \langle \text{repeat } c \text{ until } b, \sigma' \rangle \Downarrow \sigma''}{\langle \text{repeat } c \text{ until } b, \sigma \rangle \Downarrow \sigma''}$$

- Leiten Sie daraus die Rekursionsgleichung für  $\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket$  her.
- Erweitern Sie die Definition von  $\mathcal{D} \llbracket \_ \rrbracket$  um **repeat c until b**.
- Prüfen Sie, ob die Semantik mit Ihrer Erweiterung weiterhin wohldefiniert und kompositional ist.
- Zeigen oder widerlegen Sie:  $\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket = \mathcal{D} \llbracket c; \text{ while (not } b) \text{ do } c \rrbracket$

**Lösung:**

(4a) Analog zur **while**-Schleife erhält man:

$$\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket \sigma = \begin{cases} \mathcal{D} \llbracket c \rrbracket \sigma & \text{falls } \mathcal{B} \llbracket b \rrbracket (\mathcal{D} \llbracket c \rrbracket \sigma) = \mathbf{tt} \\ \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket (\mathcal{D} \llbracket c \rrbracket \sigma) & \text{falls } \mathcal{B} \llbracket b \rrbracket (\mathcal{D} \llbracket c \rrbracket \sigma) = \mathbf{ff} \end{cases}$$

Problematisch an dieser Gleichung ist, dass  $\mathcal{D} \llbracket c \rrbracket \sigma$  ggf. gar nicht definiert ist, somit die Fallunterscheidung mit  $\mathcal{B} \llbracket b \rrbracket$  ebenfalls nicht.

*Alternativer Lösungsvorschlag:*

In dieser Lösung wird das o.g. Problem umgangen, da die Funktionskomposition  $\circ$  selbst dann definiert ist, wenn  $\mathcal{D} \llbracket c \rrbracket \sigma$  undefiniert ist.

$$\begin{aligned} \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket &= \left( \lambda \sigma. \begin{cases} \sigma & \text{falls } \mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{tt} \\ \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket (\sigma) & \text{falls } \mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{ff} \end{cases} \right) \circ \mathcal{D} \llbracket c \rrbracket \\ &= \text{IF} (\mathcal{B} \llbracket b \rrbracket, \text{id}, \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket) \circ \mathcal{D} \llbracket c \rrbracket \end{aligned}$$

- (4b) Analog zur **while**-Schleife nimmt man den Fixpunkt des zur Rekursionsgleichung zugehörigen Funktionals:

$$\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket = \text{FIX}(\lambda f. \text{IF}(\mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket, \mathcal{D} \llbracket c \rrbracket, f \circ \mathcal{D} \llbracket c \rrbracket))$$

Dafür muss man aber  $\text{IF}(\_, \_, \_)$  auf partielle Prädikate  $p :: \Sigma \rightarrow \mathbb{B}$  verallgemeinern:

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

*Alternativer Lösungsvorschlag:*

In dieser Variante muss  $\text{IF}(\_, \_, \_)$  *nicht* auf partielle Prädikate erweitert werden.

$$\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket = \text{FIX}(\lambda f. \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, f) \circ \mathcal{D} \llbracket c \rrbracket)$$

- (4c) Für Wohldefiniertheit ist zu zeigen, dass das neue Funktional unter dem Fixpunkt immer kettenstetig ist, wie Thm. 109 für das Funktional von **while**.

*Beweis.* Seien also  $p :: \Sigma \Rightarrow \mathbb{B}$  und  $g :: \Sigma \rightarrow \Sigma$  beliebig. Zu zeigen ist, dass das Funktional  $F(f) = \text{IF}(p, g, f \circ g)$  monoton und kettenstetig ist.

Dazu schreibt man zuerst das Funktional wieder in einfachere Bestandteile um

$$F = (\lambda f. \text{IF}(p, g, f)) \circ (\lambda f. f \circ g)$$

In unserem Fall sind  $p = \mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket$  und  $g = \mathcal{D} \llbracket c \rrbracket$ . Nach Lem. 103 und 107 genügt es wieder, zu zeigen, dass die beiden Teile monoton und kettenstetig sind. Für den Teil  $\lambda f. f \circ g$  haben wir das schon in Thm. 109 gezeigt.

Der Beweis für  $\lambda f. \text{IF}(p, g, f)$  läuft analog zu Thm. 109, muss allerdings um den neuen Fall  $p(\sigma) = \perp$  ergänzt werden, da wir  $\text{IF}(\_, \_, \_)$  auf partielle Prädikate verallgemeinert haben.

- $\text{IF}(p, g, f)$  ist monoton in  $f$ :

Zu zeigen: Für alle  $f_1 \sqsubseteq f_2$  gilt  $\text{IF}(p, g, f_1) \sqsubseteq \text{IF}(p, g, f_2)$ . Seien also  $\sigma, \sigma'$  beliebig mit  $\text{IF}(p, g, f_1) \sigma = \sigma'$ .

– Fälle  $p(\sigma) = \mathbf{tt}$ ,  $p(\sigma) = \mathbf{ff}$ : Analog zu Beispiel 102.

– Fall  $p(\sigma) = \perp$ : Dann  $\text{IF}(p, g, f_1) \sigma = \perp$ , im Widerspruch zu  $\text{IF}(p, g, f_1) \sigma = \sigma'$ .

- $\text{IF}(p, g, f)$  ist kettenstetig in  $f$ :

Sei  $Y$  eine beliebige, nicht-leere Kette in  $(\Sigma \rightarrow \Sigma, \sqsubseteq)$ . Wegen Lem. 104 genügt es, zu zeigen:

$$\text{IF}\left(p, g, \bigsqcup Y\right) \sqsubseteq \bigsqcup \{\text{IF}(p, g, f) \mid f \in Y\}$$

Sei also  $\sigma, \sigma'$  beliebig mit  $\text{IF}(p, g, \bigsqcup Y)(\sigma) = \sigma'$ .

Zu zeigen:  $(\bigsqcup \{\text{IF}(p, g, f) \mid f \in Y\})(\sigma) = \sigma'$ .

– Fälle  $p(\sigma) = \mathbf{tt}$ ,  $p(\sigma) = \mathbf{ff}$ : Analog zu Thm. 109.

– Fall  $p(\sigma) = \perp$ :

Dann  $\text{IF}(p, g, \bigsqcup Y)(\sigma) = \perp$ , Widerspruch zu  $\text{IF}(p, g, \bigsqcup Y)(\sigma) = \sigma'$ . □

*Alternativer Lösungsvorschlag:*

Z.z. ist auch hier, dass das Funktional

$$F = (\lambda f. \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, f) \circ \mathcal{D} \llbracket c \rrbracket)$$

monoton und kettenstetig ist.

*Beweis.* Es gilt:

$$F = (\lambda f. f \circ \mathcal{D} \llbracket c \rrbracket) \circ (\lambda f. \text{IF} (\mathcal{B} \llbracket b \rrbracket, id, f))$$

Nach Bsp. 102 wissen wir, dass  $\text{IF} (p, f, g)$  monoton in  $g$  ist. Außerdem ist nach Thm. 109  $(\lambda f. f \circ g)$  monoton. Mit Lemma 103 ist daher  $F$  monoton.

Analog zu Thm. 109 lässt sich zeigen, dass  $\text{IF} (p, g, f)$  kettenstetig in  $f$  ist. Außerdem ist nach Thm. 109  $(\lambda f. f \circ g)$  kettenstetig. Mit Lemma 107 ist daher  $F$  auch kettenstetig.  $\square$

Die erweiterte Definition ist (in beiden Varianten) weiterhin kompositional, da nur die Semantiken der Teile verwendet werden. Eine nicht-kompositionale Definition wäre z.B. über die `while`-Schleife:

$$\mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket = \mathcal{D} \llbracket \text{while (not } b \text{) do } c \rrbracket \circ \mathcal{D} \llbracket c \rrbracket$$

(4d) Zuerst noch ein paar Hilfslemmata:

i.  $\text{IF} (p \circ h, f \circ h, g \circ h) = \text{IF} (p, f, g) \circ h$

*Beweis.*

$$\begin{aligned} \text{IF} (p \circ h, f \circ h, g \circ h) \sigma &= \begin{cases} (f \circ h) \sigma & \text{falls } (p \circ h)(\sigma) = \mathbf{tt} \\ (g \circ h) \sigma & \text{falls } (p \circ h)(\sigma) = \mathbf{ff} \\ \perp & \text{falls } (p \circ h)(\sigma) = \perp \end{cases} \\ &= \begin{cases} f(\sigma') & \text{falls } h(\sigma) = \sigma' \text{ und } p(\sigma') = \mathbf{tt} \\ g(\sigma') & \text{falls } h(\sigma) = \sigma' \text{ und } p(\sigma') = \mathbf{ff} \\ \perp & \text{falls } h(\sigma) = \perp \end{cases} \\ &= (\text{IF} (p, f, g) \circ h) (\sigma) \end{aligned} \quad \square$$

ii.  $\text{IF} (\mathcal{B} \llbracket \text{not } b \rrbracket, f, g) = \text{IF} (\mathcal{B} \llbracket b \rrbracket, g, f)$

*Beweis.*

$$\begin{aligned} \text{IF} (\mathcal{B} \llbracket \text{not } b \rrbracket, f, g) \sigma &= \begin{cases} f(\sigma) & \text{falls } \mathcal{B} \llbracket \text{not } b \rrbracket \sigma = \mathbf{tt} \\ g(\sigma) & \text{falls } \mathcal{B} \llbracket \text{not } b \rrbracket \sigma = \mathbf{ff} \end{cases} = \begin{cases} f(\sigma) & \text{falls } \mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{ff} \\ g(\sigma) & \text{falls } \mathcal{B} \llbracket b \rrbracket \sigma = \mathbf{tt} \end{cases} \\ &= \text{IF} (\mathcal{B} \llbracket b \rrbracket, g, f) \sigma \end{aligned} \quad \square$$

Einsetzen der Definitionen ergibt:

$$\begin{aligned} \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket &= \text{FIX} (\lambda f. \text{IF} (\mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket, \mathcal{D} \llbracket c \rrbracket, f \circ \mathcal{D} \llbracket c \rrbracket)) \\ &=: \text{FIX} (R) \\ \mathcal{D} \llbracket c; \text{while (not } b \text{) do } c \rrbracket &= \mathcal{D} \llbracket \text{while (not } b \text{) do } c \rrbracket \circ \mathcal{D} \llbracket c \rrbracket \\ &= \text{FIX} (\lambda f. \text{IF} (\mathcal{B} \llbracket \text{not } b \rrbracket, f \circ \mathcal{D} \llbracket c \rrbracket, id)) \circ \mathcal{D} \llbracket c \rrbracket \\ &=: \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket \end{aligned}$$

Um die geforderte Gleichheit zu zeigen, reicht es, folgende Approximationen zu zeigen:

$$\text{FIX} (R) \sqsubseteq \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket \quad \text{FIX} (R) \sqsupseteq \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket$$

Da  $\text{FIX}$  den kleinsten Fixpunkt berechnet, genügt es für die erste Ungleichung, zu zeigen, dass die rechte Seite ein Fixpunkt des Funktionals  $R$  ist:

$$\begin{aligned} R(\text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket) &= \text{IF} (\mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket, \mathcal{D} \llbracket c \rrbracket, (\text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket) \circ \mathcal{D} \llbracket c \rrbracket) \\ &= \text{IF} (\mathcal{B} \llbracket b \rrbracket, id, \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket) \circ \mathcal{D} \llbracket c \rrbracket \\ &= \text{IF} (\mathcal{B} \llbracket \text{not } b \rrbracket, \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket, id) \circ \mathcal{D} \llbracket c \rrbracket \\ &= W(\text{FIX} (W)) \circ \mathcal{D} \llbracket c \rrbracket = \text{FIX} (W) \circ \mathcal{D} \llbracket c \rrbracket \end{aligned}$$

Für die zweite Ungleichung schreiben wir  $\text{FIX}(R)$  noch etwas um:

$$\begin{aligned}\text{FIX}(R) &= R(\text{FIX}(R)) = \text{IF}(\mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket, \mathcal{D} \llbracket c \rrbracket, \text{FIX}(R) \circ \mathcal{D} \llbracket c \rrbracket) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket\end{aligned}$$

Da  $\circ$  monoton im ersten Argument ist (vgl. Thm. 109), reicht es für die zweite Ungleichung zu zeigen, dass:

$$\text{FIX}(W) \sqsubseteq \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R))$$

Analog zu vorher reicht es, zu zeigen, dass die rechte Seite ein Fixpunkt von  $W$  ist:

$$\begin{aligned}W(\text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R))) &= \text{IF}(\mathcal{B} \llbracket \text{not } b \rrbracket, \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket, \text{id}) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{IF}(\mathcal{B} \llbracket b \rrbracket \circ \mathcal{D} \llbracket c \rrbracket, \text{FIX}(R) \circ \mathcal{D} \llbracket c \rrbracket, \text{id} \circ \mathcal{D} \llbracket c \rrbracket)) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, R(\text{FIX}(R))) = \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R))\end{aligned}$$

Damit haben wir gezeigt, dass **repeat**-Schleifen äquivalent mächtig sind wie **while**-Schleifen, und das *ohne Induktion*, nur durch Termumschreiben – sowohl für terminierende wie auch für unendlich laufende Programme. Da  $\mathcal{D} \llbracket \_ \rrbracket$  immer noch kompositional ist, gilt diese Gleichheit auch in allen möglichen Kontexten.

*Alternativer Lösungsvorschlag:*

Wie oben setzen wir die Definitionen ein:

$$\begin{aligned}\mathcal{D} \llbracket c; \text{ while } (\text{not } b) \text{ do } c \rrbracket &= \text{FIX}(\lambda f. \text{IF}(\mathcal{B} \llbracket \text{not } b \rrbracket, f \circ \mathcal{D} \llbracket c \rrbracket, \text{id})) \circ \mathcal{D} \llbracket c \rrbracket \\ &= \text{FIX}(\lambda f. \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, f \circ \mathcal{D} \llbracket c \rrbracket)) \circ \mathcal{D} \llbracket c \rrbracket \\ &=: \text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket \\ \mathcal{D} \llbracket \text{repeat } c \text{ until } b \rrbracket &= \text{FIX}(\lambda f. \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, f) \circ \mathcal{D} \llbracket c \rrbracket) \\ &=: \text{FIX}(R)\end{aligned}$$

Es genügt es auch hier die folgenden Ungleichungen zu zeigen:

$$\text{FIX}(R) \sqsubseteq \text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket \quad \text{FIX}(R) \sqsupseteq \text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket$$

*Beweis.* Wir zeigen,  $\text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket$  ist ein Fixpunkt von  $R$ . Damit gilt die erste Ungleichung:

$$\begin{aligned}R(\text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket) &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket) \circ \mathcal{D} \llbracket c \rrbracket \\ &= W(\text{FIX}(W)) \circ \mathcal{D} \llbracket c \rrbracket \\ &= \text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket\end{aligned}$$

Außerdem gilt

$$\text{FIX}(R) = R(\text{FIX}(R)) = \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket$$

Damit lässt sich die zweite Ungleichung so umschreiben:

$$\text{FIX}(W) \circ \mathcal{D} \llbracket c \rrbracket \sqsubseteq \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket$$

Aufgrund der Monotonie von  $\circ$  (s.o.) genügt es wiederum,

$$\text{FIX}(W) \sqsubseteq \text{IF}(\mathcal{B} \llbracket b \rrbracket, \text{id}, \text{FIX}(R))$$

zu zeigen. Es gilt:

$$\begin{aligned} W(\text{IF}(\mathcal{B} \llbracket b \rrbracket, id, \text{FIX}(R))) &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, id, \text{IF}(\mathcal{B} \llbracket b \rrbracket, id, \text{FIX}(R)) \circ \mathcal{D} \llbracket c \rrbracket) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, id, R(\text{FIX}(R))) \\ &= \text{IF}(\mathcal{B} \llbracket b \rrbracket, id, \text{FIX}(R)) \end{aligned}$$

Damit ist  $\text{IF}(\mathcal{B} \llbracket b \rrbracket, id, \text{FIX}(R))$  ein Fixpunkt von  $W$  und es gilt auch die zweite Ungleichung.  $\square$