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 10: Fixpunkttheorie Besprechung: 26.06.2012

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

(a) Für $f(\sigma) = \sigma[\mathtt{i} \mapsto 1]$ und $g(\sigma) = \sigma[\mathtt{i} \mapsto 2]$ 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 Definiertheit 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 $D = \Sigma \rightharpoonup \Sigma$ und $\sqsubseteq$ die Approximationsordnung. Dann gibt es keine obere Schranke für $\{\, f,\ id \,\}$ mit $f(\sigma) = \sigma[\mathtt{x} \mapsto 1]$.

(1e) Falsch. Betrache 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 $\mathbb{R}$ exisitert). 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. 34 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 $p \neq \lambda\sigma.\ \mathtt{tt}$ gilt. Hier ein Gegenbeispiel mit $p(\sigma) = \mathtt{tt}$ und $g = id$.

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

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

2. Monotonie und Fixpunkte (H)

Finden Sie eine Halbordnung $(D, \sqsubseteq)$ mit kleinstem Element $\bot$ 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:
$D = \mathbb{N} \cup \{\,A,\ B\,\}$
$m \sqsubseteq n = m \leq n$, $m \sqsubseteq A$, $m \sqsubseteq B$, $A \sqsubseteq A$, $B \sqsubseteq B$ für alle $n, m \in \mathbb{N}$
$f(n) = n + 1$ 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. `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

$$\textsc{RepeatTT}: \frac{\langle c,\ \sigma\rangle \Downarrow \sigma' \qquad \mathcal{B}\,[\![b]\!]\,\sigma' = \mathtt{tt}}{\langle \mathtt{repeat}\ c\ \mathtt{until}\ b,\ \sigma\rangle \Downarrow \sigma'}$$

$$\textsc{RepeatFF}: \frac{\langle c,\ \sigma\rangle \Downarrow \sigma' \qquad \mathcal{B}\,[\![b]\!]\,\sigma' = \mathbf{ff} \qquad \langle \mathtt{repeat}\ c\ \mathtt{until}\ b,\ \sigma'\rangle \Downarrow \sigma''}{\langle \mathtt{repeat}\ c\ \mathtt{until}\ b,\ \sigma\rangle \Downarrow \sigma''}$$

(a) Leiten Sie daraus die Rekursionsgleichung für $\mathcal{D}\,[\![\mathtt{repeat}\ c\ \mathtt{until}\ b]\!]$ her.

(b) Erweitern Sie die Definition von $\mathcal{D}\,[\![\_]\!]$ um `repeat` c `until` b.

(c) Prüfen Sie, ob die Semantik mit Ihrer Erweiterung weiterhin wohldefiniert und kompositional ist.

(d) Zeigen oder widerlegen Sie: $\mathcal{D}\,[\![\mathtt{repeat}\ c\ \mathtt{until}\ b]\!] = \mathcal{D}\,[\![c;\ \mathtt{while}\ (\mathtt{not}\ b)\ \mathtt{do}\ c]\!]$

Lösung:

(3a) Analog zur `while`-Schleife erhält man:

$$\mathcal{D}\,[\![\mathtt{repeat}\ c\ \mathtt{until}\ b]\!]\,\sigma = \begin{cases} \mathcal{D}\,[\![c]\!]\,\sigma & \text{falls } \mathcal{B}\,[\![b]\!]\,(\mathcal{D}\,[\![c]\!]\,\sigma) = \mathtt{tt} \\ \mathcal{D}\,[\![\mathtt{repeat}\ c\ \mathtt{until}\ b]\!]\,(\mathcal{D}\,[\![c]\!]\,\sigma) & \text{falls } \mathcal{B}\,[\![b]\!]\,(\mathcal{D}\,[\![c]\!]\,\sigma) = \mathbf{ff} \end{cases}$$

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

(3b) Analog zur `while`-Schleife nimmt man den Fixpunkt des zur Rekursionsgleichung zugehörigen Funktionals:

$$\mathcal{D}\,[\![\texttt{repeat}\ c\ \texttt{until}\ b]\!] = \mathrm{FIX}\,(\lambda f.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!] \circ \mathcal{D}\,[\![c]\!],\ \mathcal{D}\,[\![c]\!],\ f \circ \mathcal{D}\,[\![c]\!]))$$

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

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

(3c) Für Wohldefiniertheit ist zu zeigen, dass das neue Funktional unter dem Fixpunkt immer kettenstetig ist, wie Thm. 34 für das Funktional von `while`.

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

Dazu schreibt man zuerst das Funktional wieder in einfachere Bestandteile um

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

In unserem Fall sind $p = \mathcal{B}\,[\![b]\!] \circ \mathcal{D}\,[\![c]\!]$ und $g = \mathcal{D}\,[\![c]\!]$. Nach Lem. 30 und 32 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. 34 gezeigt.

Der Beweis für $\lambda f.\ \mathrm{IF}\,(p,\ g,\ f)$ läuft analog zu Thm. 34, muss allerdings um den neuen Fall $p(\sigma) = \bot$ ergänzt werden, da wir $\mathrm{IF}\,(\_,\ \_,\ \_)$ auf partielle Prädikate verallgemeinert haben.

- $\mathrm{IF}\,(p,\ g,\ f)$ ist monoton in f:
  Zu zeigen: Für alle $f_1 \sqsubseteq f_2$ gilt $\mathrm{IF}\,(p,\ g,\ f_1) \sqsubseteq \mathrm{IF}\,(p,\ g,\ f_2)$. Seien also σ, σ' beliebig mit $\mathrm{IF}\,(p,\ g,\ f_1)\,\sigma = \sigma'$.
  - Fälle $p(\sigma) = \mathbf{tt}$, $p(\sigma) = \mathbf{ff}$: Analog zu Beispiel 33.
  - Fall $p(\sigma) = \bot$: Dann $\mathrm{IF}\,(p,\ g,\ f_1)\,\sigma = \bot$, im Widerspruch zu $\mathrm{IF}\,(p,\ g,\ f_1)\,\sigma = \sigma'$.
- $\mathrm{IF}\,(p,\ g,\ f)$ ist kettenstetig in f:
  Sei Y eine beliebige, nicht-leere Kette in $(\Sigma \rightharpoonup \Sigma, \sqsubseteq)$. Wegen Lem. 31 genügt es, zu zeigen:

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

  Sei also σ, σ' beliebig mit $\mathrm{IF}\,(p,\ g,\ \bigsqcup Y)\,(\sigma) = \sigma'$.
  Zu zeigen: $(\bigsqcup \{\, \mathrm{IF}\,(p,\ g,\ f) \mid f \in Y \,\})\,(\sigma) = \sigma'$.
  - Fälle $p(\sigma) = \mathbf{tt}$, $p(\sigma) = \mathbf{ff}$: Analog zu Thm. 34.
  - Fall $p(\sigma) = \bot$:
    Dann $\mathrm{IF}\,(p,\ g,\ \bigsqcup Y)\,(\sigma) = \bot$, Widerspruch zu $\mathrm{IF}\,(p,\ g,\ \bigsqcup Y)\,(\sigma) = \sigma'$. □

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

$$\mathcal{D}\,[\![\texttt{repeat}\ c\ \texttt{until}\ b]\!] = \mathcal{D}\,[\![\texttt{while (not}\ b\texttt{) do}\ c]\!] \circ \mathcal{D}\,[\![c]\!]$$

(3d) Zuerst noch ein paar Hilfslemmata:

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

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

$$= \begin{cases} f(\sigma') & \text{falls } h(\sigma) = \sigma' \text{ und } p(\sigma') = \mathbf{tt} \\ g(\sigma') & \text{falls } h(\sigma) = \sigma' \text{ und } p(\sigma') = \mathbf{ff} \\ \bot & \text{falls } h(\sigma) = \bot \end{cases}$$

$$= (\mathrm{IF}\,(p,\ f,\ g) \circ h)\,(\sigma) \qquad \square$$

ii. $\mathrm{IF}\,(\mathcal{B}\,[\![\texttt{not}\ b]\!],\ f,\ g) = \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ g,\ f)$

Beweis.

$$\mathrm{IF}\,(\mathcal{B}\,[\![\texttt{not}\ b]\!],\ f,\ g)\,\sigma = \begin{cases} f(\sigma) & \text{falls } \mathcal{B}\,[\![\texttt{not}\ b]\!]\,\sigma = \mathbf{tt} \\ g(\sigma) & \text{falls } \mathcal{B}\,[\![\texttt{not}\ b]\!]\,\sigma = \mathbf{ff} \end{cases} = \begin{cases} f(\sigma) & \text{falls } \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{ff} \\ g(\sigma) & \text{falls } \mathcal{B}\,[\![b]\!]\,\sigma = \mathbf{tt} \end{cases}$$

$$= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ f,\ g)\,\sigma \qquad \square$$

Einsetzen der Definitionen ergibt:

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

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

$$\mathrm{FIX}\,(R) \sqsubseteq \mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!] \qquad \mathrm{FIX}\,(R) \sqsupseteq \mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!]$$

Da FIX den kleinsten Fixpunkt berechnet, genügt es für die erste Umgleichung, zu zeigen, dass die rechte Seite ein Fixpunkt des Funktionals R ist:

$$\begin{aligned} R(\mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!]) &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!] \circ \mathcal{D}\,[\![c]\!],\ \mathcal{D}\,[\![c]\!],\ (\mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!]) \circ \mathcal{D}\,[\![c]\!]) \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!]) \circ \mathcal{D}\,[\![c]\!] \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![\texttt{not}\ b]\!],\ \mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!],\ id) \circ \mathcal{D}\,[\![c]\!] \\ &= W(\mathrm{FIX}\,(W)) \circ \mathcal{D}\,[\![c]\!] = \mathrm{FIX}\,(W) \circ \mathcal{D}\,[\![c]\!] \end{aligned}$$

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

$$\begin{aligned} \mathrm{FIX}\,(R) = R(\mathrm{FIX}\,(R)) &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!] \circ \mathcal{D}\,[\![c]\!],\ \mathcal{D}\,[\![c]\!],\ \mathrm{FIX}\,(R) \circ \mathcal{D}\,[\![c]\!]) \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R)) \circ \mathcal{D}\,[\![c]\!] \end{aligned}$$

Da $\circ$ monoton im ersten Argument ist (vgl. Thm. 34), reicht es für die zweite Ungleichung zu zeigen, dass:

$$\mathrm{FIX}\,(W) \sqsubseteq \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R))$$

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

$$\begin{aligned} W(\mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R))) &= \mathrm{IF}\,(\mathcal{B}\,[\![\texttt{not}\ b]\!],\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R)) \circ \mathcal{D}\,[\![c]\!],\ id) \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R)) \circ \mathcal{D}\,[\![c]\!]) \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!] \circ \mathcal{D}\,[\![c]\!],\ \mathrm{FIX}\,(R) \circ \mathcal{D}\,[\![c]\!],\ id \circ \mathcal{D}\,[\![c]\!])) \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ R(\mathrm{FIX}\,(R))) = \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ id,\ \mathrm{FIX}\,(R)) \end{aligned}$$

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 $\mathcal{D}\,[\![\_]\!]$ immer noch kompositional ist, gilt diese Gleichheit auch in allen möglichen Kontexten.