

Semantik von Programmiersprachen – SS 2017

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

Lösungen zu Blatt 9: Denotationale Semantik mit Fixpunktiteration

Besprechung: 26.06.2017

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

(a) Die folgende Rekursionsgleichung definiert die Funktion $f :: \mathbb{N} \Rightarrow \mathbb{N}$ eindeutig:

$$f(n) = \begin{cases} 2 \cdot f\left(\frac{n}{2}\right) & \text{falls } n \text{ gerade} \\ f(n+1) - 1 & \text{falls } n \text{ ungerade} \end{cases}$$

(b) Die folgende Rekursionsgleichung definiert die Funktion $g :: \mathbb{N} \Rightarrow \mathbb{N}$ eindeutig:

$$g(n) = \begin{cases} \left(g\left(\frac{n}{2}\right)\right)^2 & \text{falls } n \text{ gerade} \\ g(n-1) & \text{falls } n \text{ ungerade} \end{cases}$$

(c) Das Funktional $F :: (\Sigma \rightarrow \Sigma) \Rightarrow (\Sigma \rightarrow \Sigma)$ mit

$$F(f) = \lambda\sigma. \begin{cases} (f(\sigma[x \mapsto 42]))[y \mapsto 23] & \text{falls } \sigma(z) \leq 17 \\ \sigma[z \mapsto 0] & \text{sonst} \end{cases}$$

gehört zur Rekursionsgleichung

$$Q(\sigma) = \begin{cases} (Q(\sigma[x \mapsto 42]))[y \mapsto 23] & \text{falls } \sigma(z) \leq 17 \\ \sigma[z \mapsto 0] & \text{sonst} \end{cases}$$

(d) Für das Funktional $F :: (\Sigma \rightarrow \Sigma) \Rightarrow (\Sigma \rightarrow \Sigma)$ mit

$$F(f) = \lambda\sigma. \begin{cases} f(\sigma[y \mapsto 12]) & \text{falls } \sigma(x) = 13 \\ \sigma[x \mapsto 10] & \text{sonst} \end{cases}$$

gilt

$$F^{42}(\perp)(\sigma) = \begin{cases} \perp & \text{falls } \sigma(x) = 13 \\ \sigma[x \mapsto 10] & \text{sonst} \end{cases}$$

(e) $\mathcal{D}[[c_1; (c_2; c_3)]] = \mathcal{D}[[c_2; c_1]; c_3]$

(f) $\mathcal{D}[[\text{if } (x \leq y) \text{ then } z := x \text{ else } z := y]; \text{skip}]] \sigma = \sigma[z \mapsto \min(\sigma(x), \sigma(y))]$

Lösung:

- (1a) Richtig. Man muss dafür zeigen, dass es eine Lösung gibt und dass die Rekursion irgendwann terminiert.

Die Identitätsfunktion id ist eine Lösung:

$$n = id(n) = \begin{cases} 2 \cdot \frac{n}{2} = 2 \cdot id\left(\frac{n}{2}\right) & \text{falls } n \text{ gerade} \\ (n+1) - 1 = id(n+1) - 1 & \text{falls } n \text{ ungerade} \end{cases}$$

Damit ist klar, dass die Rekursionsgleichung erfüllbar ist.

Für die Eindeutigkeit muss man nur noch beweisen, dass wenn f die Gleichung für alle n erfüllt, dann gilt $f(n) = n$ für alle n . Beweis per vollständiger Induktion über n :

Sei also n beliebig. Induktionsannahme: Für alle $m < n$ gilt: $f(m) = m$. Zu zeigen: $f(n) = n$.

- Fall $n = 0$: Aus der Rekursionsgleichung folgt: $f(0) = 2 \cdot f\left(\frac{0}{2}\right) = 2 \cdot f(0)$ und damit $f(0) = 0$.
- Fall $n = 1$: Dann gilt entsprechend: $f(1) = f(1+1) - 1 = 2 \cdot f(1) - 1$ und damit $f(1) = 1$.
- Fall $n > 1$ gerade: Dann ist $\frac{n}{2} < n$, also folgt aus der Induktionsannahme mit $m = \frac{n}{2}$, dass $f\left(\frac{n}{2}\right) = \frac{n}{2}$. Damit gilt $f(n) = 2 \cdot f\left(\frac{n}{2}\right) = 2 \cdot \frac{n}{2} = n$.
- Fall $n > 1$ ungerade: Dann ist $\frac{n+1}{2} < n$, also folgt aus der Induktionsannahme mit $m = \frac{n+1}{2}$, dass $f\left(\frac{n+1}{2}\right) = \frac{n+1}{2}$. Damit gilt: $f(n) = f(n+1) - 1 = 2 \cdot f\left(\frac{n+1}{2}\right) - 1 = 2 \cdot \frac{n+1}{2} - 1 = n$.

Damit ist id die einzige Funktion, die die Rekursionsgleichung erfüllt.

- (1b) Falsch. Die konstanten Funktionen $g_1(n) = 0$ und $g_2(n) = 1$ erfüllen beide die Rekursionsgleichung.

- (1c) Richtig.

- (1d) Richtig. Es gilt:

$$F^0(\perp)(\sigma) = \perp$$

$$F^1(\perp)(\sigma) = F(\perp)(\sigma) = \begin{cases} \perp & \text{falls } \sigma(\mathbf{x}) = 13 \\ \sigma[\mathbf{x} \mapsto 10] & \text{sonst} \end{cases}$$

$$F^2(\perp)(\sigma) = F(F^1(\perp))(\sigma) = \begin{cases} F^1(\perp)(\sigma[\mathbf{y} \mapsto 12]) = \perp & \text{falls } \sigma(\mathbf{x}) = 13 \\ \sigma[\mathbf{x} \mapsto 10] & \text{sonst} \end{cases} = F^1(\perp)$$

Entsprechend $F^{42}(\perp) = F^{41}(\perp) = \dots = F^1(\perp)$

- (1e) Falsch. Gegenbeispiel $c_1 = \mathbf{x} := 1$, $c_2 = \mathbf{x} := 0$, $c_3 = \text{skip}$. Dann gilt:

$$\begin{aligned} \mathcal{D}[[c_1; (c_2; c_3)]] &= \mathcal{D}[[c_2; c_3]] \circ \mathcal{D}[[c_1]] = (\mathcal{D}[[c_3]] \circ \mathcal{D}[[c_2]]) \circ \mathcal{D}[[c_1]] = \lambda\sigma. \sigma[\mathbf{x} \mapsto 0] \\ \mathcal{D}[(c_2; c_1); c_3] &= \mathcal{D}[[c_3]] \circ \mathcal{D}[[c_2; c_1]] = \mathcal{D}[[c_3]] \circ (\mathcal{D}[[c_1]] \circ \mathcal{D}[[c_2]]) = \lambda\sigma. \sigma[\mathbf{x} \mapsto 1] \end{aligned}$$

Richtig wäre $\mathcal{D}[[c_1; (c_2; c_3)]] = \mathcal{D}[[c_1; c_2]; c_3]$. Zum Beweis einfach ausrechnen und die Assoziativität von \circ ausnutzen.

- (1f) Richtig. Sei c das ganze Programm, if der if-Teil.

$$\begin{aligned} \mathcal{D}[[P]] \sigma &= (\mathcal{D}[[\text{skip}]] \circ \mathcal{D}[[if]]) (\sigma) = (id \circ \mathcal{D}[[if]]) (\sigma) = \mathcal{D}[[if]] \sigma \\ &= \text{IF}(\mathcal{B}[\mathbf{x} \leq \mathbf{y}], \mathcal{D}[[z := \mathbf{x}]], \mathcal{D}[[z := \mathbf{y}]]) \sigma \\ &= \text{IF}(\lambda\sigma. \sigma(\mathbf{x}) \leq \sigma(\mathbf{y}), \lambda\sigma. \sigma[\mathbf{z} \mapsto \sigma(\mathbf{x})], \lambda\sigma. \sigma[\mathbf{z} \mapsto \sigma(\mathbf{y})]) \sigma \\ &= \begin{cases} \sigma[\mathbf{z} \mapsto \sigma(\mathbf{x})] & \text{falls } \sigma(\mathbf{x}) \leq \sigma(\mathbf{y}) \\ \sigma[\mathbf{z} \mapsto \sigma(\mathbf{y})] & \text{sonst} \end{cases} \\ &= \sigma[\mathbf{z} \mapsto \min(\sigma(\mathbf{x}), \sigma(\mathbf{y}))] \end{aligned}$$

2. Fixpunktiteration (H)

Gegeben sei folgendes Programm P :

```
x := 0; i := n; while (1 <= i) do (x := x + 2 * i; i := i - 1)
```

- Geben Sie das Funktional F an, das in der denotationellen Semantik zur Schleife gehört.
- Berechnen Sie $F^0(\perp)$, $F^1(\perp)$, $F^2(\perp)$ und $F^3(\perp)$.
- Geben Sie $F^n(\perp)$ und $\text{FIX}(F)$ an (ohne Beweis).
- Geben Sie $\mathcal{D} \llbracket P \rrbracket$ an.

Lösung:

(2a) Zuerst die Semantik des Schleifenrumpfs:

$$\begin{aligned} \mathcal{D} \llbracket x := x + 2 * i; i := i - 1 \rrbracket \sigma &= (\mathcal{D} \llbracket i := i - 1 \rrbracket \circ \mathcal{D} \llbracket x := x + 2 * i \rrbracket)(\sigma) \\ &= \mathcal{D} \llbracket i := i - 1 \rrbracket (\sigma[x \mapsto \sigma(x) + 2 \cdot \sigma(i)]) \\ &= \sigma[x \mapsto \sigma(x) + 2 \cdot \sigma(i), i \mapsto \sigma(i) - 1] \end{aligned}$$

Damit ergibt sich das Funktional F zu:

$$\begin{aligned} F(f)(\sigma) &= \text{IF}(\mathcal{B} \llbracket 1 \leq i \rrbracket, f \circ \mathcal{D} \llbracket x := x + 2 * i; i := i - 1 \rrbracket, id) \sigma \\ &= \text{IF}(\lambda \sigma. 1 \leq \sigma(i), \lambda \sigma. f(\sigma[x \mapsto \sigma(x) + 2 \cdot \sigma(i), i \mapsto \sigma(i) - 1]), id) \sigma \\ &= \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ f(\sigma[x \mapsto \sigma(x) + 2 \cdot \sigma(i), i \mapsto \sigma(i) - 1]) & \text{falls } 1 \leq \sigma(i) \end{cases} \end{aligned}$$

(2b) Sei $\sigma' = \sigma[x \mapsto \sigma(x) + 2 \cdot \sigma(i), i \mapsto \sigma(i) - 1]$.

$$F^0(\perp)(\sigma) = \perp$$

$$F^1(\perp)(\sigma) = F(F^0(\perp))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ F^0(\perp)(\sigma') = \perp & \text{falls } 1 \leq \sigma(i) \end{cases}$$

$$F^2(\perp)(\sigma) = F(F^1(\perp))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ F^1(\perp)(\sigma') & \text{falls } 1 \leq \sigma(i) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma' & \text{falls } 1 \leq \sigma(i) \text{ und } 1 > \sigma'(i) \\ \perp & \text{falls } 1 \leq \sigma(i) \text{ und } 1 \leq \sigma'(i) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma[x \mapsto \sigma(x) + 2, i \mapsto 0] & \text{falls } 1 = \sigma(i) \\ \perp & \text{falls } 1 < \sigma(i) \end{cases}$$

$$F^3(\perp)(\sigma) = F(F^2(\perp))(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ F^2(\perp)(\sigma') & \text{falls } 1 \leq \sigma(i) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma' & \text{falls } 1 \leq \sigma(i) \text{ und } 1 > \sigma'(i) \\ \sigma'[x \mapsto \sigma'(x) + 2, i \mapsto 0] & \text{falls } 1 \leq \sigma(i) \text{ und } 1 = \sigma'(i) \\ \perp & \text{falls } 1 \leq \sigma(i) \text{ und } 1 < \sigma'(i) \end{cases}$$

$$= \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma[x \mapsto \sigma(x) + 2, i \mapsto 0] & \text{falls } 1 = \sigma(i) \\ \sigma[x \mapsto \sigma(x) + 6, i \mapsto 0] & \text{falls } 2 = \sigma(i) \\ \perp & \text{falls } 2 < \sigma(i) \end{cases}$$

(2c)

$$F^n(\perp)(\sigma) = \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma[x \mapsto \sigma(i) \cdot (\sigma(i) + 1) + \sigma(x), i \mapsto 0] & \text{falls } \sigma(i) \in \{1, \dots, n\} \\ \perp & \text{falls } n < \sigma(i) \end{cases}$$
$$\text{FIX}(F)\sigma = \begin{cases} \sigma & \text{falls } 1 > \sigma(i) \\ \sigma[x \mapsto \sigma(i) \cdot (\sigma(i) + 1) + \sigma(x), i \mapsto 0] & \text{falls } 1 \leq \sigma(i) \end{cases}$$

(2d) Sei $w = \text{while } (1 \leq i) \text{ do } (x := x + 2 * i; i := i - 1)$

$$\begin{aligned} \mathcal{D} \llbracket P \rrbracket \sigma &= (\mathcal{D} \llbracket w \rrbracket \circ \mathcal{D} \llbracket i := n \rrbracket \circ \mathcal{D} \llbracket x := 0 \rrbracket)(\sigma) = (\mathcal{D} \llbracket w \rrbracket \circ \mathcal{D} \llbracket i := n \rrbracket)(\sigma[x \mapsto 0]) \\ &= \mathcal{D} \llbracket w \rrbracket (\sigma[x \mapsto 0, i \mapsto \sigma(n)]) = \text{FIX}(F)(\sigma[x \mapsto 0, i \mapsto \sigma(n)]) \\ &= \begin{cases} \sigma[x \mapsto 0, i \mapsto \sigma(n)] & \text{falls } \sigma(n) < 1 \\ \sigma[x \mapsto \sigma(n) \cdot (\sigma(n) + 1), i \mapsto 0] & \text{falls } \sigma(n) \geq 1 \end{cases} \end{aligned}$$

3. Lokale Variablen in Blöcken (Ü)

In Kap. 6.3 haben wir die While-Sprache um Blöcke mit lokalen Variablen erweitert und operationale Semantiken dafür angegeben. Erweitern Sie in diesem Abschnitt die denotationale Semantik entsprechend.

- (a) Definieren Sie $\mathcal{D} \llbracket _ \rrbracket$ für das neue Sprachkonstrukt $\{ \text{var } x = a; c \}$. Bleibt dadurch $\mathcal{D} \llbracket _ \rrbracket$ kompositional?
- (b) Zeigen oder widerlegen Sie:

$$\mathcal{D} \llbracket \{ \text{var } z = x; x := y; y := z \} \rrbracket = \mathcal{D} \llbracket x := x + y; y := x - y; x := x - y \rrbracket$$

Lösung:

- (3a) Letztendlich ist ein Block nichts anderes als die Zuweisung des Startwerts an x , der Ausführung von c und einer erneuten Zuweisung (des ursprünglichen Werts von x) an x . Damit ergibt sich:

$$\begin{aligned} \mathcal{D} \llbracket \{ \text{var } x = a; c \} \rrbracket \sigma &= ((\lambda \sigma'. \sigma'[x \mapsto \sigma(x)] \circ \mathcal{D} \llbracket c \rrbracket \circ (\lambda \sigma'. \sigma'[x \mapsto \mathcal{A} \llbracket a \rrbracket \sigma]))(\sigma) \\ &= (\mathcal{D} \llbracket c \rrbracket (\sigma[x \mapsto \mathcal{A} \llbracket a \rrbracket \sigma]))[x \mapsto \sigma(x)] \end{aligned}$$

wobei das äußere Zustandsupdate undefiniertheit propagieren soll. Die denotationale Semantik bleibt dadurch kompositional, da nur die Bedeutungen $\mathcal{A} \llbracket a \rrbracket$ und $\mathcal{D} \llbracket c \rrbracket$ der Teile a und c verwendet werden.

- (3b) Beweis:

$$\begin{aligned} &\mathcal{D} \llbracket \{ \text{var } z = x; x := y; y := z \} \rrbracket \sigma \\ &= (\mathcal{D} \llbracket x := y; y := z \rrbracket (\sigma[z \mapsto \sigma(x)]))[z \mapsto \sigma(z)] \\ &= (\mathcal{D} \llbracket y := z \rrbracket \circ \mathcal{D} \llbracket x := y \rrbracket)(\sigma[z \mapsto \sigma(x)])[z \mapsto \sigma(z)] \\ &= (\mathcal{D} \llbracket y := z \rrbracket (\sigma[z \mapsto \sigma(x), x \mapsto (\sigma[z \mapsto \sigma(x)](y))]))[z \mapsto \sigma(z)] \\ &= (\mathcal{D} \llbracket y := z \rrbracket (\sigma[z \mapsto \sigma(x), x \mapsto \sigma(y)]))[z \mapsto \sigma(z)] \\ &= \sigma[z \mapsto \sigma(x), x \mapsto \sigma(y), y \mapsto (\sigma[z \mapsto \sigma(x), x \mapsto \sigma(y)](z)), z \mapsto \sigma(z)] \\ &= \sigma[x \mapsto \sigma(y), y \mapsto \sigma(x), z \mapsto \sigma(z)] = \sigma[x \mapsto \sigma(y), y \mapsto \sigma(x)] \end{aligned}$$

$$\begin{aligned}
& \mathcal{D} \llbracket x := x + y; y := x - y; x := x - y \rrbracket \sigma \\
&= (\mathcal{D} \llbracket x := x - y \rrbracket \circ \mathcal{D} \llbracket y := x - y \rrbracket \circ \mathcal{D} \llbracket x := x + y \rrbracket) \sigma \\
&= (\mathcal{D} \llbracket x := x - y \rrbracket \circ \mathcal{D} \llbracket y := x - y \rrbracket) (\sigma[x \mapsto \sigma(x) + \sigma(y)]) \\
&= \mathcal{D} \llbracket x := x - y \rrbracket (\sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto (\sigma[x \mapsto \sigma(x) + \sigma(y)])(x) - (\sigma[x \mapsto \sigma(x) + \sigma(y)])(y)]) \\
&= \mathcal{D} \llbracket x := x - y \rrbracket (\sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto (\sigma(x) + \sigma(y)) - \sigma(y)]) \\
&= \mathcal{D} \llbracket x := x - y \rrbracket (\sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto \sigma(x)]) \\
&= \sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto \sigma(x), \\
&\quad x \mapsto (\sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto \sigma(x)])(x) - (\sigma[x \mapsto \sigma(x) + \sigma(y), y \mapsto \sigma(x)])(y)] \\
&= \sigma[y \mapsto \sigma(x), x \mapsto (\sigma(x) + \sigma(y)) - \sigma(x)] = \sigma[y \mapsto \sigma(x), x \mapsto \sigma(y)]
\end{aligned}$$