

Semantik von Programmiersprachen – SS 2017

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

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.

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.

3. Exkurs: monadische Semantik (H)

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

$$\begin{aligned} \text{pure} &:: a \Rightarrow m a \\ (>=>) &:: (a \Rightarrow m b) \Rightarrow (b \Rightarrow m c) \Rightarrow (a \Rightarrow m c) \end{aligned}$$

Statt auf $\Sigma \rightarrow \Sigma$ können wir $\mathcal{D}[_]$ auf dem allgemeineren Typ $\Sigma \Rightarrow m \Sigma$ definieren, angenommen $m \Sigma$ ist eine ccpo¹.

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

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

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

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

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

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

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