Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen
Sebastian Ullrich sebastian.ullrich@kit.edu
Max Wagner maximilian.wagner@kit.edu

Semantik von Programmiersprachen – SS 2019

http://pp.ipd.kit.edu/lehre/SS2019/semantik

Blatt 10: Fixpunkttheorie Besprechung: 01.06.2019

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) $\mathrm{IF}\ (p,\ f,\ g)$ 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 $\bot$ und eine monotone Funktion $f :: D \to D$, die mehrere Fixpunkte besitzt, aber keinen kleinsten.

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:

$$\begin{aligned} \mathit{pure} &:: a \to m\ a \\ (>\!\!=\!\!>) &:: (a \to m\ b) \to (b \to m\ c) \to (a \to m\ c) \end{aligned}$$

Statt auf $\Sigma \rightharpoonup \Sigma$ können wir $\mathcal{D}\,[\![\_]\!]$ auf dem allgemeineren Typ $\Sigma \to m\ \Sigma$ definieren, angenommen $m\ \Sigma$ ist eine ccpo[1].

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

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

[1] Man überlege sich, dass $\Sigma \to m\ \Sigma$ 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

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

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

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

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

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

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