Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen

Denis Lohner denis.lohner@kit.edu

Sebastian Ullrich sebastian.ullrich@kit.edu

Semantik von Programmiersprachen – SS 2017

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

Lösungen zu Blatt 12: Continuations Besprechung: 17.07.2017

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

(a) `raise X; x := y - z; try skip catch X skip` ist ein While_X-Programm.

(b) `raise X` und `while (true) do skip` sind semantisch äquivalent.

(c) Alle partiellen Korrektheitseigenschaften sind zulässig.

(d) Strikte Funktionen $\varphi :: (D, \sqsubseteq) \Rightarrow (\mathbb{B}, \leq)$ (wobei $\mathbf{ff} \leq \mathbf{tt}$) sind zulässige Prädikate

Lösung:

(1a) Richtig. `raise` X-Aufrufe müssen nicht innerhalb von `try _ catch` X `_`-Blöcken stehen.

(1b) Das hängt von der Semantikdefinition ab, die man nimmt. In den Big-Step- und Small-Step-Semantiken der Vorlesung sind sie es nicht:

$$\langle \texttt{raise } X,\ (\texttt{None}, \sigma)\rangle \Downarrow (X, \sigma) \qquad \text{aber} \qquad \langle \texttt{while (true) do skip},\ (\texttt{None}, \sigma)\rangle \not\Downarrow$$

$$\langle \texttt{raise } X,\ \sigma\rangle \xrightarrow{*}_1 \langle \texttt{raise } X,\ \sigma\rangle \not\to_1 \qquad \text{aber} \qquad \langle \texttt{while (true) do skip},\ \sigma\rangle \xrightarrow{\infty}_1$$

In der Fortsetzungssemantik der Vorlesung auch erst einmal nicht:

$$\mathcal{C}[\![\texttt{raise } X]\!]\, sS\sigma = S(X)(\sigma) \qquad \text{aber} \qquad \mathcal{C}[\![\texttt{while (true) do skip}]\!]\, sS\sigma = \bot$$

Nimmt man aber für S die anfängliche Fortsetzung $S_0(X)(\sigma) = \bot$, so sind sie gleich.

(1c) Richtig – obwohl hier vordergründig Begriffe aus verschiedenen Semantiken vermischt werden. Eine partielle Korrektheitseigenschaft φ hat die Form: Für alle $f :: \Sigma \rightharpoonup \Sigma$ und σ, σ' mit $f(\sigma) = \sigma'$ folgt aus $P(\sigma)$, dass $Q(\sigma')$ gilt.
Sei also Y Kette in unserer Approximationsordnung $(\Sigma \rightharpoonup \Sigma, \sqsubseteq)$.
Zu zeigen: Wenn $\varphi(f)$ für alle $f \in Y$, dann gilt auch $\varphi(\bigsqcup Y)$.
Seien also σ und σ' beliebig mit $\bigsqcup Y(\sigma) = \sigma'$ und $P(\sigma)$. Zu zeigen: $Q(\sigma')$.
Wegen $\bigsqcup Y(\sigma) = \sigma'$ gibt es nach Definition ein $f \in Y$ mit $f(\sigma) = \sigma'$. Damit $\varphi(f)$ und wegen $P(\sigma)$ folgt daraus $Q(\sigma')$.

(1d) Falsch. Strikte Funktionen müssen auch die Schranke von leeren Ketten erhalten:

$$\varphi\left(\bigsqcup \emptyset\right) = \bigsqcup \varphi(\emptyset) = \bigsqcup \emptyset = \mathbf{ff}$$

Andererseits ist $\emptyset$ eine Kette mit $\varphi(d)$ für alle $d \in \emptyset$. Zulässigkeit würde also $\varphi(\bigsqcup \emptyset) = \mathbf{tt}$ verlangen.
Entsprechend ist auch Kettenstetigkeit im Allgemeinen nicht mit Zulässigkeit vergleichbar: $\varphi(d) = \mathbf{ff}$ ist kettenstetig, aber nicht zulässig. Umgekehrt sei $D = \mathbb{N}^\infty$ mit der natürlichen Ordnung und

$$\varphi(n) = (n = 0 \vee n = \infty)$$

Dann ist φ zulässig, aber nicht einmal monoton!

2. Kompositionalität der Fortsetzungsemantik (H)

In dieser Aufgabe sollen die Kompositionalitätsbetrachtungen für $\mathcal{D}[\![\_]\!]$ auf die Fortsetzungssemantik $\mathcal{C}[\![\_]\!]$ übertragen werden.

(a) Erweitern Sie die Definition der Kontexte auf While_X. Passen Sie die Kontextfüllfunktion $\_\,[\_]$ an die Erweiterung an.

(b) Definieren Sie die Fortsetzungssemantik $\overline{\mathcal{K}}[\![K]\!]$ eines Kontexts K analog zu $\mathcal{K}[\![K]\!]$.

(c) Formulieren Sie ein Kompositionalitätstheorem für $\mathcal{C}[\![\_]\!]$ und $\overline{\mathcal{K}}[\![\_]\!]$ analog zu Thm. 115. Beweisen Sie es.

Lösung:

(2a) Man braucht zwei neue Kontextproduktionen für $\texttt{try}\ c\ \texttt{catch}\ X\ c'$, da dies zwei Teilanweisungen enthalten kann. Für $\texttt{raise}\ X$ gibt es *keine* neue Kontextproduktion:
$\mathsf{Context}\quad K ::= \ldots \mid \texttt{try}\ K\ \texttt{catch}\ X\ c \mid \texttt{try}\ c\ \texttt{catch}\ X\ K$
Entsprechend die Definition für $\_\,[\_]$:

$$\begin{aligned}(\texttt{try}\ K\ \texttt{catch}\ X\ c)\,[c'] &= \texttt{try}\ K\,[c']\ \texttt{catch}\ X\ c\\ (\texttt{try}\ c\ \texttt{catch}\ X\ K)\,[c'] &= \texttt{try}\ c\ \texttt{catch}\ X\ K\,[c']\end{aligned}$$

(2b) Die Semantikfunktion $\overline{\mathcal{K}}[\![\_]\!]$ hat nun den Typ

$$\mathsf{Context} \Rightarrow (\mathsf{Cont} \Rightarrow (\mathsf{Xcp} \Rightarrow \mathsf{Cont}) \Rightarrow \mathsf{Cont}) \Rightarrow (\mathsf{Cont} \Rightarrow (\mathsf{Xcp} \Rightarrow \mathsf{Cont}) \Rightarrow \mathsf{Cont})$$

entsprechend dem Typ $\mathsf{Com} \Rightarrow \mathsf{Cont} \Rightarrow (\mathsf{Xcp} \Rightarrow \mathsf{Cont}) \Rightarrow \mathsf{Cont}$ der Semantikfunktion $\mathcal{C}[\![\_]\!]$. Die Definition für $\overline{\mathcal{K}}[\![\_]\!]$ folgt dem Muster der Definition von $\mathcal{C}[\![\_]\!]$, man ersetzt $\mathcal{C}[\![c]\!]$ durch $\overline{\mathcal{K}}[\![K]\!]\,f$ an den Stellen, die dem Kontext entsprechen.

$$\begin{aligned}\overline{\mathcal{K}}[\![[]]\!]\,f\ s\ S &= f(s)(S)\\ \overline{\mathcal{K}}[\![K\texttt{;}\ c]\!]\,f\ s\ S &= \overline{\mathcal{K}}[\![K]\!]\,f\ (\mathcal{C}[\![c]\!]\ s\ S)\ S\\ \overline{\mathcal{K}}[\![c\texttt{;}\ K]\!]\,f\ s\ S &= \mathcal{C}[\![c]\!]\,(\overline{\mathcal{K}}[\![K]\!]\,f\ s\ S)\ S\\ \overline{\mathcal{K}}[\![\texttt{if (}b\texttt{) then }K\texttt{ else }c]\!]\,f\ s\ S &= \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ \overline{\mathcal{K}}[\![K]\!]\,f\ s\ S,\ \mathcal{C}[\![c]\!]\ s\ S)\\ \overline{\mathcal{K}}[\![\texttt{if (}b\texttt{) then }c\texttt{ else }K]\!]\,f\ s\ S &= \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ \mathcal{C}[\![c]\!]\ s\ S,\ \overline{\mathcal{K}}[\![K]\!]\,f\ s\ S)\\ \overline{\mathcal{K}}[\![\texttt{while (}b\texttt{) do }K]\!]\,f\ s\ S &= \mathrm{FIX}\,(\lambda g.\ \mathrm{IF}\,(\mathcal{B}[\![b]\!],\ \overline{\mathcal{K}}[\![K]\!]\,f\ g\ S,\ s))\\ \overline{\mathcal{K}}[\![\texttt{try }K\texttt{ catch }X\ c]\!]\,f\ s\ S &= \overline{\mathcal{K}}[\![K]\!]\,f\ s\ (S[X \mapsto \mathcal{C}[\![c]\!]\ s\ S])\\ \overline{\mathcal{K}}[\![\texttt{try }c\texttt{ catch }X\ K]\!]\,f\ s\ S &= \mathcal{C}[\![c]\!]\ s\ (S[X \mapsto \overline{\mathcal{K}}[\![K]\!]\,f\ s\ S])\end{aligned}$$

(2c) Kompositionalitätsaussage:

$$\mathcal{C}[\![K\,[c']]\!]\ s\ S = \overline{\mathcal{K}}[\![K]\!]\,(\mathcal{C}[\![c']\!])s\ S$$

Beweis. Analog zu Thm. 115 durch Induktion über K (s und S beliebig) und ausrechnen:
- Fall $[]$: $\mathcal{C}[\![[]\,[c']]\!]\ s\ S = \mathcal{C}[\![c']\!]\ s\ S = \overline{\mathcal{K}}[\![[]]\!]\,(\mathcal{C}[\![c']\!])s\ S$
- Fall $K\texttt{;}\ c$:

$$\begin{aligned}\mathcal{C}[\![(K\texttt{;}\ c)\,[c']]\!]\ s\ S &= \mathcal{C}[\![K\,[c']\texttt{;}\ c]\!]\ s\ S = \mathcal{C}[\![K\,[c']]\!]\,(\mathcal{C}[\![c]\!]\ s\ S)\ S\\ &\overset{IH}{=} \overline{\mathcal{K}}[\![K]\!]\,(\mathcal{C}[\![c']\!])(\mathcal{C}[\![c]\!]\ s\ S)\ S = \overline{\mathcal{K}}[\![K\texttt{;}\ c]\!]\,(\mathcal{C}[\![c']\!])\ s\ S\end{aligned}$$

- Fall $c\texttt{;}\ K$:

$$\begin{aligned}\mathcal{C}[\![(c\texttt{;}\ K)\,[c']]\!]\ s\ S &= \mathcal{C}[\![c\texttt{;}\ (K\,[c'])]\!]\ s\ S = \mathcal{C}[\![c]\!]\,(\mathcal{C}[\![K\,[c']]\!]\ s\ S)\ S\\ &\overset{IH}{=} \mathcal{C}[\![c]\!]\,(\overline{\mathcal{K}}[\![K]\!]\,(\mathcal{C}[\![c']\!])\ s\ S)\ S = \overline{\mathcal{K}}[\![c\texttt{;}\ K]\!]\,(\mathcal{C}[\![c']\!])\ s\ S\end{aligned}$$

- Fall `if (`b`) then` K `else` c:

$$\begin{aligned}\mathcal{C}\,[\![\texttt{(if (}b\texttt{) then }K\texttt{ else }c\texttt{)}\,[c']]\!]\ s\ S &= \mathcal{C}\,[\![\texttt{if (}b\texttt{) then }K\,[c']\texttt{ else }c]\!]\ s\ S = \\ &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \mathcal{C}\,[\![K\,[c']]\!]\ s\ S,\ \mathcal{C}\,[\![c]\!]\,s\ S) \\ &\overset{IH}{=} \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \overline{\mathcal{K}}\,[\![K]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S,\ \mathcal{C}\,[\![c]\!]\,s\ S) \\ &= \overline{\mathcal{K}}\,[\![\texttt{if (}b\texttt{) then }K\texttt{ else }c]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S\end{aligned}$$

- Fall `if (`b`) then` c `else` K: Analog.
- Fall `while (`b`) do` K:

$$\begin{aligned}\mathcal{C}\,[\![\texttt{(while (}b\texttt{) do }K\texttt{)}\,[c']]\!]\,s\ S &= \mathcal{C}\,[\![\texttt{while (}b\texttt{) do (}K\,[c']\texttt{)}]\!]\,s\ S \\ &= \mathrm{FIX}\,(\lambda g.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \mathcal{C}\,[\![K\,[c']]\!]\,g\ S,\ s)) \\ &\overset{IH}{=} \mathrm{FIX}\,(\lambda g.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \overline{\mathcal{K}}\,[\![K]\!]\,(\mathcal{C}\,[\![c']\!])\ g\ S,\ s)) \\ &= \overline{\mathcal{K}}\,[\![\texttt{while (}b\texttt{) do }K]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S\end{aligned}$$

- Fall `try` K `catch` X c:

$$\begin{aligned}\mathcal{C}\,[\![\texttt{(try }K\texttt{ catch }X\ c\texttt{)}\,[c']]\!]\,s\ S &= \mathcal{C}\,[\![\texttt{try }K\,[c']\texttt{ catch }X\ c]\!]\,s\ S \\ &= \mathcal{C}\,[\![K\,[c']]\!]\,s\ (S[X \mapsto \mathcal{C}\,[\![c]\!]\,s\ S]) \\ &\overset{IH}{=} \overline{\mathcal{K}}\,[\![K]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ (S[X \mapsto \mathcal{C}\,[\![c]\!]\,s\ S]) \\ &= \overline{\mathcal{K}}\,[\![\texttt{try }K\texttt{ catch }X\ c]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S\end{aligned}$$

- Fall `try` c `catch` X K:

$$\begin{aligned}\mathcal{C}\,[\![\texttt{(try }c\texttt{ catch }X\ K\texttt{)}\,[c']]\!]\,s\ S &= \mathcal{C}\,[\![\texttt{try }c\texttt{ catch }X\ K\,[c']]\!]\,s\ S \\ &= \mathcal{C}\,[\![c]\!]\,s\ (S[X \mapsto \mathcal{C}\,[\![K\,[c']]\!]\,s\ S]) \\ &\overset{IH}{=} \mathcal{C}\,[\![c]\!]\,s\ (S[X \mapsto \overline{\mathcal{K}}\,[\![K]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S]) \\ &= \overline{\mathcal{K}}\,[\![\texttt{try }c\texttt{ catch }X\ K]\!]\,(\mathcal{C}\,[\![c']\!])\ s\ S\end{aligned}$$

□

3. Semantik für ASM (Ü)

In Kap. 5.1 wurde eine Small-Step-Semantik für die idealisierte Assembler-Sprache ASM angegeben. In dieser Aufgabe sollen Sie nun eine denotationale Fortsetzungssemantik für ASM angeben. Dazu soll jeder Indexposition in der Instruktionsliste P eine Fortsetzung zugeordnet werden, die in einer Umgebung $\mathsf{IEnv} = \mathbb{Z} \Rightarrow (\Sigma \rightharpoonup \Sigma)$ gespeichert werden.

(a) Geben Sie eine Funktion $\mathcal{I}\,[\![\_,\ \_]\!] :: (\mathbb{Z} \times \mathsf{Asm}) \Rightarrow \mathsf{IEnv} \Rightarrow (\Sigma \rightharpoonup \Sigma)$ mit folgender Bedeutung an: Sind i eine Indexposition, I eine Instruktion und E eine Umgebung mit Fortsetzungen, dann beschreibt $\mathcal{I}\,[\![i,\ I]\!]\,E$ die Ausführung der Instruktion I mit Indexposition i zusammen mit der Forsetzung aus $E(n)$, wobei n die Indexposition der Instruktion angibt, die nach I auszuführen wäre.

(b) Geben Sie ein Funktional $F :: \mathsf{Asm}\ \mathit{list} \Rightarrow \mathsf{IEnv} \Rightarrow \mathsf{IEnv}$ an, das für eine Instruktionsliste P und einer Fortsetzungsumgebung E dieses E wie folgt erweitert: $F[\![P]\!]E$ enthält für jede Instruktionsposition n die Fortsetzung, die zuerst die Instruktion an Stelle n ausführt und danach mit der entsprechenden Fortsetzung aus E fortfährt. Verwenden Sie dazu die Funktion $\mathcal{I}\,[\![\_,\ \_]\!]$.

(c) Geben Sie die Semantik $[\![P]\!]$ einer Instruktionsliste P an, wobei $[\![\_]\!] :: \mathsf{Asm}\ \mathit{list} \Rightarrow (\Sigma \rightharpoonup \Sigma)$. Verwenden Sie dazu den kleinsten Fixpunkt des Funktionals F.

(d) Wie könnte man die Existenz und Eindeutigkeit dieses kleinsten Fixpunkts beweisen?

(e) Ist diese Semantik kompositional? Diskutieren Sie!

Lösung:

(3a) Definition von $\mathcal{I}[\![i,\ I]\!]\,E$ durch Fallunterscheidung über Asm:

$$\begin{array}{lcl} \mathcal{I}[\![i,\ \texttt{ASSN}\ x\ a]\!]\,E\,\sigma & = & E(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) \\ \mathcal{I}[\![i,\ \texttt{JMP}\ k]\!]\,E\,\sigma & = & E(i+k)\sigma \\ \mathcal{I}[\![i,\ \texttt{JMPF}\ k\ b]\!]\,E\,\sigma & = & \begin{cases} E(i+1)\sigma & \text{falls } \mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt} \\ E(i+k)\sigma & \text{falls } \mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff} \end{cases} \end{array}$$

(3b)

$$F[\![P]\!]E\ n = \begin{cases} \mathcal{I}[\![n,\ P_n]\!]\,E & \text{falls } 0 \le n < |P| \\ id & \text{sonst} \end{cases}$$

(3c)

$$[\![P]\!] = \mathrm{FIX}\,(F[\![P]\!])\,(0)$$

(3d) Entsprechend der Fixpunkttheorie aus dem Skript: IEnv muss eine ccpo sein und das Funktional $F[\![P]\!]$ monoton und kettenstetig.

Nach Lemma 122 ist $\mathsf{IEnv} = \mathbb{Z} \Rightarrow (\Sigma \rightharpoonup \Sigma)$ eine ccpo bezüglich der punktweisen Ordnung $\sqsubseteq'$.

$F[\![P]\!]$ hat den Typ $\mathsf{IEnv} \Rightarrow \mathsf{IEnv}$, ist also für die Fixpunktberechnung geeignet. Für den Fixpunktsatz muss man also noch zeigen, dass $F[\![P]\!]$ monoton und kettenstetig ist.

- $\mathcal{I}[\![i,\ I]\!]$ ist monoton.
  Beweis. Sei $E_1 \sqsubseteq' E_2$. Zu zeigen: $\mathcal{I}[\![i,\ I]\!]\,E_1 \sqsubseteq \mathcal{I}[\![i,\ I]\!]\,E_2$.
  Seien also σ und σ' mit $\mathcal{I}[\![i,\ I]\!]\,E_1\sigma = \sigma'$ Zu zeigen: $\mathcal{I}[\![i,\ I]\!]\,E_2\sigma = \sigma'$. Fallunterscheidung nach I:
  - Fall $I = \texttt{ASSN}\ x\ a$: Es gilt $\mathcal{I}[\![i,\ I]\!]\,E_1\sigma = E_1(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) = \sigma'$. Wegen $E_1 \sqsubseteq' E_2$ gilt $E_1(i+1) \sqsubseteq E_2(i+1)$ und damit auch $E_2(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) = \sigma'$. Damit nach Definition $\mathcal{I}[\![i,\ I]\!]\,E_2\sigma = \sigma'$.
  - Fall $I = \texttt{JMPF}\ k\ b$: Fallunterscheidung nach $\mathcal{B}[\![b]\!]\,\sigma$:
    - Fall $\mathbf{tt}$: Es gilt $\mathcal{I}[\![i,\ I]\!]\,E_1\sigma = E_1(i+1)\sigma = \sigma'$. Wegen $E_1 \sqsubseteq' E_2$ ist $E_1(i+1) \sqsubseteq E_2(i+1)$ und somit $E_2(i+1)\sigma = \sigma'$, also $\mathcal{I}[\![i,\ I]\!]\,E_2\sigma = \sigma'$.
    - Fall $\mathbf{ff}$: Es gilt $\mathcal{I}[\![i,\ I]\!]\,E_1\sigma = E_1(i+k)\sigma = \sigma'$. Wegen $E_1 \sqsubseteq' E_2$ ist $E_1(i+k) \sqsubseteq E_2(i+k)$ und somit $E_2(i+k)\sigma = \sigma'$, also $\mathcal{I}[\![i,\ I]\!]\,E_2\sigma = \sigma'$.
  - Fall $I = \texttt{JMP}\ k$: Analog. □
- $F[\![P]\!]$ ist monoton.
  Beweis. Sei $E_1 \sqsubseteq' E_2$. Zu zeigen: $F[\![P]\!]E_1 \sqsubseteq' F[\![P]\!]E_2$. Sei also $n \in \mathbb{Z}$ beliebig. Für $0 \le n < |P|$ gilt

  $$F[\![P]\!]E_1\ n = \mathcal{I}[\![n,\ P_n]\!]\,E_1 \sqsubseteq \mathcal{I}[\![n,\ P_n]\!]\,E_2 = F[\![P]\!]E_2\ n$$

  und für sonstige n gilt

  $$F[\![P]\!]E_1\ n = id \sqsubseteq id = F[\![P]\!]E_2\ n$$ □
- $\mathcal{I}[\![i,\ I]\!]$ ist kettenstetig.
  Beweis. Sei Y nichtleere Kette in $(\mathsf{IEnv}, \sqsubseteq')$. Zu zeigen:

  $$\mathcal{I}[\![i,\ I]\!]\left(\bigsqcup{}' Y\right) \sqsubseteq \bigsqcup \{\, \mathcal{I}[\![i,\ I]\!]\,E \mid E \in Y \,\}$$

  Sei also σ, σ' mit $\mathcal{I}[\![i,\ I]\!]\,(\bigsqcup' Y)\sigma = \sigma'$. Zu zeigen: $(\bigsqcup \{\, \mathcal{I}[\![i,\ I]\!]\,E \mid E \in Y \,\})(\sigma) = \sigma'$.
  Fallunterscheidung nach I:

- Fall $I = \texttt{ASSN}\ x\ a$: Dann gilt $\mathcal{I}[\![i,\ I]\!]\left(\bigsqcup' Y\right)\sigma = \left(\bigsqcup' Y\right)(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) = \sigma'$. Nach Definition von $\bigsqcup' Y$ folgt: $\left(\bigsqcup \{\, E(i+1) \mid E \in Y \,\}\right)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) = \sigma'$. Damit gibt es also ein $E \in Y$ mit $E(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) = \sigma'$ nach Definition von $\bigsqcup\_$ auf $(\Sigma \rightharpoonup \Sigma, \sqsubseteq)$.
  Wegen $E \in Y$ ist $\lambda\sigma.\ E(i+1)(\sigma[x \mapsto \mathcal{A}[\![a]\!]\,\sigma]) \sqsubseteq \bigsqcup \{\, \mathcal{I}[\![i,\ I]\!]\,E \mid E \in Y \,\}$, somit folgt die Behauptung.
- Fälle $I = \texttt{JMPF}\ k\ b$ und $I = \texttt{JMP}\ k$: Entsprechend. □

- $F[\![P]\!]$ ist kettenstetig.
  Beweis. Sei Y nichtleere Kette in $(\mathsf{IEnv}, \sqsubseteq')$.
  Zu zeigen: $F[\![P]\!]\left(\bigsqcup' Y\right) \sqsubseteq' \bigsqcup \{\, F[\![P]\!]E \mid E \in Y \,\}$
  Sei $n \in \mathbb{Z}$ beliebig. Nach Definition zu zeigen: $F[\![P]\!]\left(\bigsqcup' Y\right)n \sqsubseteq \left(\bigsqcup \{\, F[\![P]\!]E \mid E \in Y \,\}\right)(n)$.
  Es gilt:

$$F[\![P]\!]\left(\bigsqcup' Y\right)n = \begin{cases} \mathcal{I}[\![n,\ P_n]\!]\left(\bigsqcup' Y\right) & \text{falls } 0 \le n < |P| \\ id & \text{sonst} \end{cases}$$
$$\sqsubseteq \begin{cases} \bigsqcup \{\mathcal{I}[\![n,\ P_n]\!]\,E \mid E \in Y \,\} \\ id \end{cases} \quad = \left(\bigsqcup \{\, F[\![P]\!]E \mid E \in Y \,\}\right)(n) \qquad \square$$

(3e) Kompositionalität hängt vom Verständnis der Komponenten bzw. des Kontexts ab. Sieht man als Komponenten eines Programms die arithmetischen und booleschen Formeln an, so ist die Definition offensichtlich kompositional, da lediglich die Semantikfunktionen $\mathcal{A}[\![\_]\!]$ und $\mathcal{B}[\![\_]\!]$ verwendet werden.

Die Semantikdefinition ist jedoch *nicht* kompositional in Bezug auf Listenkonkatenation: Die Semantik des Programms $P +\!+ P'$ lässt sich nicht aus der Semantik von P und P' rekonstruieren – schon gar nicht, wenn man $[\![\_]\!]$ als Semantik zu Grunde legt, da diese jegliche Information über die Semantik der einzelnen Inzides außer dem ersten verwirft. Dies entspricht der Einschränkung der Continuation-Semantik für While_X auf feste Start-Fortsetzungen, für die auch keine Kompositionalität besteht (siehe oben).

Auch beim uneingeschränkten Fixpunkt $\mathrm{FIX}\,(F[\![P]\!])$ hat man keine direkte Kompositionalität, da das Funktional $F[\![P]\!]$ bei Sprüngen an Stellen außerhalb der Instruktionsliste einfach aufhört. Wenn man allerdings zwei abgeschlossene Programme hintereinandersetzt, sollte $\mathrm{FIX}\,(F[\![P]\!])$ kompositional sein, ganz enstprechend zu den Verschiebung- und Aufteilungsslemmata bei der Compilerverifikation.

Über Kompositionalität im Sinne der While-Sprache zu reden, macht an dieser Stelle keinen Sinn, da man dann die Komponentendiskussion auf vom Compiler generierte Programme einschränken müsste, die eine ganz spezielle Form haben.