6.5 Continuation-style denotationale Semantik

In Kapitel 5.4 haben wir bereits die Erweiterung While_X von While um Ausnahmen und deren Behandlung mit den Anweisungen `raise` X und `try` c_1 `catch` X c_2 betrachtet. Bei der Big-Step-Semantik haben wir ein zusätzliches Rückgabeflag eingeführt, um normale und außergewöhnliche Termination zu unterscheiden. Entsprechend mussten auch alle bestehenden Regeln sorgfältig angepasst und die Möglichkeit für eine Ausnahme in jedem Schritt eigens behandelt werden. In der Small-Step-Semantik musste dazu eine eigene `raise` _-Regel für alle zusammengesetzten Konstrukte (bei uns nur c_1 `;` c_2) eingeführt werden.

Ganz analog zur Big-Step-Semantik ließe sich auch die denotationale Semantik für While um Exceptions erweitern. Allerdings ist dieser Formalismus insgesamt nicht zufrieden stellend, da Ausnahmen nun einmal die Ausnahme sein und deswegen nicht explizit durch jedes Programmkonstrukt durchgeschleift werden sollten. Dafür gibt es Fortsetzungen (continuations), die die Semantik (d.h. den Effekt) der Ausführung des restlichen Programms beschreiben. Eine einfache Form der Fortsetzungen haben wir schon in der Übung zur Goto-Erweiterung bei der Small-Step-Semantik gesehen, bei der aber die Fortsetzung noch syntaktisch als Anweisungsliste repräsentiert war.

Definition 43 (Fortsetzung, continuation). Eine *Fortsetzung* (*continuation*) ist eine partielle Funktion auf Zuständen, die das Ergebnis der Ausführung des restlichen Programms von einem Zustand aus beschreibt. $\mathsf{Cont} = \Sigma \rightharpoonup \Sigma$ bezeichne die Menge aller Fortsetzungen.[5]

Statt nun anhand eines Flags einem umgebenden Programmkonstrukts die Auswahl der restlichen Berechnung zu überlassen, soll jedes Konstrukt direkt auswählen, ob es normal oder mit einer (und welcher) Ausnahmebehandlung weitergehen soll. Dazu muss die Semantik der restlichen normalen bzw. außergewöhnlichen Auswertung direkt bei der Definition eines Konstrukts als Fortsetzung zur Verfügung stehen. Zum Beispiel wählt `raise` X die Fortsetzung „Ausnahmebehandlung für X“ und `skip` die Fortsetzung „normale Ausführung“ aus. Alle möglichen Fortsetzungen müssen also als Parameter an die Semantikfunktion $\mathcal{C}\,[\![\_]\!]$ gegeben werden, die damit den Typ

$$\mathsf{Com} \Rightarrow \underbrace{\mathsf{Cont}}_{\text{normale Fortsetzung}} \Rightarrow (\underbrace{\mathsf{Xcp} \Rightarrow \mathsf{Cont}}_{\text{Ausnahmebehandlung}}) \Rightarrow \mathsf{Cont}$$

hat. Intuitiv bedeutet $\mathcal{C}\,[\![c]\!]\ s\ S\ \sigma$ also: Führe c im Zustand σ aus und setze mit s normal bzw. mit $S(X)$ bei der Ausnahme X fort. S ist dabei die “Umgebung” der Ausnahmebehandlungen, die von `try` _ `catch` _ _ blockartig geändert wird. Formal:

$$\begin{aligned}
\mathcal{C}\,[\![\texttt{skip}]\!]\ s\ S &= s \\
\mathcal{C}\,[\![x\ \texttt{:=}\ a]\!]\ s\ S &= \lambda\sigma.\ s(\sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma]) \\
\mathcal{C}\,[\![c_1\texttt{;}\ c_2]\!]\ s\ S &= \mathcal{C}\,[\![c_1]\!]\,(\mathcal{C}\,[\![c_2]\!]\ s\ S)\ S \\
\mathcal{C}\,[\![\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2]\!]\ s\ S &= \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \mathcal{C}\,[\![c_1]\!]\ s\ S,\ \mathcal{C}\,[\![c_2]\!]\ s\ S) \\
\mathcal{C}\,[\![\texttt{while (}b\texttt{) do}\ c]\!]\ s\ S &= \mathrm{FIX}\,(\lambda f.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \mathcal{C}\,[\![c]\!]\ f\ S,\ s)) \\
\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ s\ S &= S(X) \\
\mathcal{C}\,[\![\texttt{try}\ c_1\ \texttt{catch}\ X\ c_2]\!]\ s\ S &= \mathcal{C}\,[\![c_1]\!]\ s\ (S[X \mapsto \mathcal{C}\,[\![c_2]\!]\ s\ S])
\end{aligned}$$

Für ein Programm verwendet man üblicherweise die anfänglichen Fortsetzungen $s_0 = id$ und $S_0 = \bot$, sofern man Nichttermination und unbehandelte Ausnahmen nicht unterscheiden möchte. Ansonsten muss man Cont auf eine allgemeinere Antwortmenge wie z.B. $\Sigma \rightharpoonup (\mathsf{Xcp}' \times \Sigma)$ bei der Big-Step-Semantik verallgemeinern – in diesem Fall wären dann $s_0(\sigma) = (\texttt{None}, \sigma)$ und $S_0(X)(\sigma) = (X, \sigma)$.

[5] Genau genommen ist es nicht wesentlich, was der Rückgabetyp einer Fortsetzung selbst ist. In unserem Fall werden das meist Zustände (Σ) sein, im Allgemeinen nimmt man aber oft den abstrakten Typ *Answer*.

Beispiel 36. Sei $c =$ `try (x := 2; raise Xcpt; x := 3) catch Xcpt x := 4`.

$$\begin{aligned}
\mathcal{C}\,[\![c]\!]\,s\;S\;\sigma &= \mathcal{C}\,[\![\texttt{x := 2; raise Xcpt; x := 3}]\!]\,s\,(\overbrace{S[\texttt{Xcpt} \mapsto \mathcal{C}\,[\![\texttt{x := 4}]\!]\,s\;S]}^{=S'})\,\sigma \\
&= \mathcal{C}\,[\![\texttt{x := 2}]\!]\,(\mathcal{C}\,[\![\texttt{raise Xcpt; x := 3}]\!]\,s\;S')\;S'\;\sigma \\
&= \mathcal{C}\,[\![\texttt{raise Xcpt; x := 3}]\!]\,s\;S'\;(\sigma[\texttt{x} \mapsto 2]) \\
&= \mathcal{C}\,[\![\texttt{raise Xcpt}]\!]\,(\mathcal{C}\,[\![\texttt{x := 3}]\!]\,s\;S')\;S'\;(\sigma[\texttt{x} \mapsto 2]) \\
&= S'(\texttt{Xcpt})\;(\sigma[\texttt{x} \mapsto 2]) = \mathcal{C}\,[\![\texttt{x := 4}]\!]\,s\;S\;(\sigma[\texttt{x} \mapsto 2]) = s(\sigma[\texttt{x} \mapsto 4])
\end{aligned}$$

Damit gilt für $s = s_0 = id$ und $S = S_0 = \bot$: $\mathcal{C}\,[\![c]\!]\,s\;S\;\sigma = \mathcal{C}\,[\![c]\!]\,id\;\bot\;\sigma = \sigma[\texttt{x} \mapsto 4]$.

Noch ein paar Anmerkungen zur Continuation-Semantik $\mathcal{C}\,[\![\_]\!]$:

- $\mathcal{C}\,[\![\texttt{skip}]\!]$ ist nicht mehr einfach die Identität, sondern die Fortsetzung. Das tritt analog auch bei `while (`b`) do` c auf.
- Die Reihenfolge von c_1 und c_2 in $\mathcal{C}\,[\![c_1\texttt{;}\;\; c_2]\!]$ ist nicht mehr wie bei $\mathcal{D}\,[\![c_1\texttt{;}\;\; c_2]\!]$ vertauscht.
- Das Funktional für den Fixpunktoperator in der Gleichung `while (`b`) do` c entstammt der Rekursionsgleichung

  $$\mathcal{C}\,[\![\texttt{while (}b\texttt{) do }c]\!]\,s\;S = \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\;\mathcal{C}\,[\![c]\!]\,(\mathcal{C}\,[\![\texttt{while (}b\texttt{) do }c]\!]\,s\;S)\;S,\;\;s)$$

  Es ist dabei implizit von den Parametern s und S abhängig: Sein kleinster Fixpunkt definiert $\mathcal{C}\,[\![\texttt{while (}b\texttt{) do }c]\!]\,s\;S$ nur für feste s und S.

Wie in Kap. 6.3 muss man nun noch nachweisen, dass FIX wirklich definiert ist. Dies ist diesmal aber aufwändiger, weil der Parameter f des Funktionals der Semantikfunktion $\mathcal{C}\,[\![c]\!]$ des Schleifenrumpfs c als Parameter übergeben wird, d.h., Monotonie und Kettenstetigkeit des Funktionals erfordern, dass $\mathcal{C}\,[\![c]\!]$ selbst *monoton* und *kettenstetig* ist. Dafür benötigen wir einige Vorbereitungen:

Lemma 40 (Diagonalregel). Sei $(D, \sqsubseteq_D)$ eine Halbordnung, $(E, \sqsubseteq_E)$ eine ccpo und sei $f :: D \Rightarrow D \Rightarrow E$ monoton in beiden Parametern. Dann gilt für alle Ketten Y:

$$\bigsqcup\nolimits_E \{ \bigsqcup\nolimits_E \{ f(d_1)(d_2) \mid d_2 \in Y \} \mid d_1 \in Y \} = \bigsqcup\nolimits_E \{ f(d)(d) \mid d \in Y \}$$

Beweis.

- Alle vorkommenden $\bigsqcup$ existieren: Da f monoton ist, zeigt man leicht, dass alle vorkommenden Mengen Ketten sind. Da E eine ccpo ist, existieren auch die kleinsten oberen Schranken.
- $\sqsubseteq_E$: Es genügt zu zeigen, dass die rechte Seite eine obere Schranke von $\{ \bigsqcup_E \{ f(d_1)(d_2) \mid d_2 \in Y \} \mid d_1 \in Y \}$ ist. Dafür genügt es zu zeigen, dass für alle $d_1 \in Y$ die rechte Seite eine obere Schranke von $\{ f(d_1)(d_2) \mid d_2 \in Y \}$ ist.

  Seien also $d_1, d_2 \in Y$. Da Y eine Kette ist, gilt $d_1 \sqsubseteq_D d_2$ oder $d_2 \sqsubseteq_D d_1$. ObdA. gelte $d_1 \sqsubseteq_D d_2$ – der andere Fall ist symmetrisch. Dann gilt: $f(d_1)(d_2) \sqsubseteq_E f(d_2)(d_2) \sqsubseteq_E \bigsqcup_E \{ f(d)(d) \mid d \in Y \}$.
- $\sqsupseteq_E$: Es genügt zu zeigen, dass die linke Seite eine obere Schranke von $\{ f(d)(d) \mid d \in Y \}$ ist.

  Sei also $d \in Y$. Dann gilt:

  $$f(d)(d) \sqsubseteq_E \bigsqcup\nolimits_E \{ f(d)(d_2) \mid d_2 \in Y \} \sqsubseteq_E \bigsqcup\nolimits_E \{ \bigsqcup\nolimits_E \{ f(d_1)(d_2) \mid d_2 \in Y \} \mid d_1 \in Y \} \qquad \square$$

Die Diagonalregel erlaubt, mehrere obere Schranken über die gleiche Kette zu einer oberen Schranke zusammenzufassen.

Definition 44 (Punktweise Halbordnung). Sei $(D, \sqsubseteq)$ eine Halbordnung. Die punktweise Halbordnung $\sqsubseteq'$ auf $A \Rightarrow D$ ist definiert durch $f \sqsubseteq' g$ gdw. $\forall a \in A.\ f(a) \sqsubseteq g(a)$

Lemma 41. $(A \Rightarrow D, \sqsubseteq')$ ist eine Halbordnung. Wenn $(D, \sqsubseteq)$ eine ccpo ist, so ist $(A \Rightarrow D, \sqsubseteq')$ auch eine ccpo mit

$$\left(\bigsqcup{}' Y\right)(n) = \bigsqcup \{\, f(n) \mid f \in Y \,\}$$

Beweis. Der Beweis, dass $(A \Rightarrow D, \sqsubseteq')$ ist eine Halbordnung, ist trivial.

Sei also $(D, \sqsubseteq)$ eine ccpo und Y eine Kette in $(A \Rightarrow D, \sqsubseteq')$.

- $\bigsqcup' Y$ ist wohldefinitert: $Y(a) = \{\, f(a) \mid f \in Y \,\}$ ist eine Kette in $(D, \sqsubseteq)$.

  Sei $d_1, d_2 \in Y(a)$. Dann gibt es $f_1, f_2 \in Y$ mit $f_1(a) = d_1$ und $f_2(a) = d_2$. Da Y eine Kette ist, gilt $f_1 \sqsubseteq' f_2$ oder $f_2 \sqsubseteq' f_1$. Damit nach Definition aber $f_1(a) \sqsubseteq' f_2(a)$ oder $f_2(a) \sqsubseteq' f_1(a)$.
- $\bigsqcup' Y$ ist eine obere Schranke von Y: Sei $f \in Y$. Zu zeigen: $f \sqsubseteq' \bigsqcup Y$.

  Nach Definition ist für beliebiges $a \in A$ zu zeigen, dass $f(a) \sqsubseteq (\bigsqcup Y)(a)$. Wegen $f \in Y$ ist $f(a) \in Y(a)$. Damit also $f(a) \sqsubseteq \bigsqcup (Y(a)) = \left(\bigsqcup' Y\right)(a)$.
- $\bigsqcup' Y$ ist die kleinste obere Schranke von Y: Sei f obere Schranke von Y. Zu zeigen: $\bigsqcup' Y \sqsubseteq' f$.

  Nach Definition ist für alle $a \in A$ zu zeigen, dass $\left(\bigsqcup' Y\right)(a) \sqsubseteq f(a)$. Wegen $\left(\bigsqcup' Y\right)(a) = \bigsqcup (Y(a))$ genügt es zu zeigen, dass $f(a)$ obere Schranke von $Y(a)$ ist.

  Sei $d \in Y(a)$ beliebig. Dann gibt es ein $f_0 \in Y$ mit $f_0(a) = d$. Da f obere Schranke von Y ist, gilt $f_0 \sqsubseteq' f$, also $d = f_0(a) \sqsubseteq f(a)$ nach Definition von $\sqsubseteq'$. □

In Fall der Ausnahmefortsetzungsumgebungen verwenden wir die Ordnung $\sqsubseteq'$ für $A = \mathsf{Xcp}$ und $D = \mathsf{Cont}$.

Lemma 42 (Monotonie und Kettenstetigkeit der Ausnahmeumgebungsaktualisierung). Die Funktion $f(S)(s) = S[X \mapsto s]$ für festes X ist monoton und kettenstetig in beiden Argumenten.

Beweis. Nachrechnen! □

Lemma 43. Seien $(D, \sqsubseteq_D)$, $(E, \sqsubseteq_E)$ und $(F, \sqsubseteq_F)$ ccpos. Seien $f :: D \Rightarrow E \Rightarrow F$ und $g :: D \Rightarrow E$ monoton in allen Argumenten. Dann ist auch $h(d) = f(d)(g(d))$ monoton. Sind f und g zusätzlich kettenstetig in allen Argumenten, so ist auch h kettenstetig.

Beweis. Monotonie: Sei $d_1 \sqsubseteq_D d_2$. Dann gilt $g(d_1) \sqsubseteq_E g(d_2)$ wegen der Monotonie von g. Mit der Monotonie von f folgt dann: $h(d_1) = f(d_1)(g(d_1)) \sqsubseteq_F f(d_2)(g(d_1)) \sqsubseteq_F f(d_2)(g(d_2)) = h(d_2)$.

Kettenstetigkeit: Sei Y nicht-leere Kette in D. Dann gilt wegen der Kettenstetigkeit von f und g:

$$\begin{aligned} h(\bigsqcup Y) &= f(\bigsqcup Y)(g(\bigsqcup Y)) = \bigsqcup \{\, f(d)(g(\bigsqcup Y)) \mid d \in Y \,\} \\ &= \bigsqcup \{\, f(d)(\bigsqcup \{\, g(d') \mid d' \in Y \,\}) \mid d \in Y \,\} = \bigsqcup \{\, \bigsqcup \{\, f(d)(g(d')) \mid d' \in Y \,\} \mid d \in Y \,\} = \ldots \end{aligned}$$

Die Funktion $h'(d)(d') = f(d)(g(d'))$ ist monoton, also folgt mit der Diagonalregel (Lem. 40):

$$\ldots = \bigsqcup \{\, f(d)(g(d)) \mid d \in Y \,\} = \bigsqcup \{\, h(d) \mid d \in Y \,\}$$ □

Korollar 44. Beliebige Kombinationen monotoner bzw. kettenstetiger Funktionen durch Funktionsanwendung sind monoton bzw. kettenstetig.

Beweis. Kombiniert man kettenstetiger Funktionen nur durch Funktionsanwendung und Parameter, so kann man diese Kombination aus mit den SKI-Kombinatoren der kombinatorischen Logik schreiben. Lem. 43 zeigt, dass der S-Kombinator kettenstetig ist. Die Kombinatoren K (konstante Funktion) und I (Identität) sind trivialerweise kettenstetig. □

Lem. 43 und Kor. 44 verallgemeinern Lem. 30 und Lem. 32 auf beliebige Kompositionen monotoner bzw. kettenstetiger Funktionen.

Lemma 45 (FIX ist monoton). Sei $(D, \sqsubseteq)$ eine ccpo und seien $f_1 \sqsubseteq' f_2$ monoton und kettenstetig. Dann ist $\mathrm{FIX}\,(f_1) \sqsubseteq \mathrm{FIX}\,(f_2)$.

Beweis. Wegen $\mathrm{FIX}\,(f_1) = \bigsqcup \{\, f_1^n(\bot) \mid n \in \mathbb{N} \,\}$ genügt es zu zeigen, dass $\mathrm{FIX}\,(f_2)$ obere Schranke von $\{\, f_1^n(\bot) \mid n \in \mathbb{N} \,\}$ ist. Sei also $n \in \mathbb{N}$. Mittels Induktion über n erhält man $f_1^n(\bot) \sqsubseteq f_2^n(\bot)$:

- Fall n = 0: $f_1^0(\bot) = \bot \sqsubseteq \bot = f_2^0(\bot)$.
- Fall n + 1: Induktionsannahme: $f_1^n(\bot) \sqsubseteq f_2^n(\bot)$.
  Zusammen mit der Monotonie von f_1 und $f_1 \sqsubseteq' f_2$ gilt:

$$f_1^{n+1}(\bot) = f_1(f_1^n(\bot)) \sqsubseteq f_1(f_2^n(\bot)) \sqsubseteq f_2(f_2^n(\bot)) = f_2^{n+1}(\bot)$$

Wegen $\mathrm{FIX}\,(f_2) = \bigsqcup \{\, f_2^n(\bot) \mid n \in \mathbb{N} \,\}$ gilt also $f_1^n(\bot) \sqsubseteq f_2^n(\bot) \sqsubseteq \mathrm{FIX}\,(f_2)$, d. h., $\mathrm{FIX}\,(f_2)$ ist obere Schranke und $\mathrm{FIX}\,(f_1)$ als kleinste obere Schranke ist kleiner oder gleich $\mathrm{FIX}\,(f_2)$. □

Lemma 46 (FIX ist kettenstetig). Sei $(D, \sqsubseteq)$ eine ccpo und Z eine nicht-leere Kette in $(D \Rightarrow D, \sqsubseteq')$, die nur monotone und kettenstetige Funktionen enthält. Dann ist $\{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}$ eine nicht-leere Kette in $(D, \sqsubseteq)$, $\bigsqcup' Z$ ist monoton und kettenstetig, und es gilt:

$$\mathrm{FIX}\left(\bigsqcup' Z\right) = \bigsqcup \{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}$$

Beweis.

- $\{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}$ ist nicht-leere Kette: Da FIX monoton ist und alle $f \in Z$ monoton und kettenstetig sind (Lem. 45), folgt dies aus der Kettenerhaltung (Lem. 31).
- $\bigsqcup' Z$ ist monoton: Sei $d_1 \sqsubseteq d_2$. Wegen $\left(\bigsqcup' Z\right)(d_1) = \bigsqcup \{\, f(d_1) \mid f \in Z \,\}$ genügt es zu zeigen, dass $\left(\bigsqcup' Z\right)(d_2)$ eine obere Schranke von $\{\, f(d_1) \mid f \in Z \,\}$ ist, da $\left(\bigsqcup' Z\right)(d_1)$ die kleinste solche ist.
  Sei also $f \in Z$. Da f monoton ist, gilt $f(d_1) \sqsubseteq f(d_2) \sqsubseteq \bigsqcup \{\, f(d_2) \mid f \in Z \,\} = \left(\bigsqcup' Z\right)(d_2)$.
- $\bigsqcup' Z$ ist kettenstetig: Sei Y nicht-leere Kette in $(D, \sqsubseteq)$. Nach Lem. 31 genügt es zu zeigen, dass $\left(\bigsqcup' Z\right)\left(\bigsqcup Y\right) \sqsubseteq \bigsqcup \{\, \left(\bigsqcup' Z\right)(d) \mid d \in Y \,\}$.
  Da alle $f \in Z$ kettenstetig sind, gilt

$$\begin{aligned}\left(\bigsqcup' Z\right)\left(\bigsqcup Y\right) &= \bigsqcup \{\, f(\bigsqcup Y) \mid f \in Z \,\} = \bigsqcup \{ \bigsqcup \{\, f(d) \mid d \in Y \,\} \mid f \in Z \,\} \\ &\sqsubseteq \bigsqcup \{ \bigsqcup \{\, f(d) \mid f \in Z \,\} \mid d \in Y \,\} = \bigsqcup \{\, \left(\bigsqcup' Z\right)(d) \mid d \in Y \,\}\end{aligned}$$

  wobei die Vertauschung der Limiten im $\sqsubseteq$-Schritt einfach nachzurechnen ist.
- FIX ist kettenstetig:
  Wir zeigen die Gleichheit über Antisymmetrie:
  - $\sqsubseteq$: Da $\mathrm{FIX}\left(\bigsqcup' Z\right)$ kleinster Fixpunkt von $\bigsqcup' Z$ ist, genügt es zu zeigen, dass die rechte Seite ein Fixpunkt von $\bigsqcup' Z$ ist.
    Da alle $g \in Z$ kettenstetig sind, gilt:

$$\begin{aligned}\left(\bigsqcup' Z\right)\left(\bigsqcup \{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}\right) &= \bigsqcup \left\{\, g\left(\bigsqcup \{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}\right) \mid g \in Z \,\right\} \\ &= \bigsqcup \left\{ \bigsqcup \{\, g(\mathrm{FIX}\,(f)) \mid f \in Z \,\} \mid g \in Z \,\right\} = \ldots\end{aligned}$$

    Da FIX monoton ist (Lem. 45), lässt sich die Diagonalregel anwenden, um die Schranken zusammenzufassen:

$$\ldots = \bigsqcup \{\, f(\mathrm{FIX}\,(f)) \mid f \in Z \,\} = \bigsqcup \{\, \mathrm{FIX}\,(f) \mid f \in Z \,\}$$

– $\sqsupseteq$: Hier genügt es zu zeigen, dass $\mathrm{FIX}\left(\bigsqcup' Z\right)$ eine obere Schranke von $\{\,\mathrm{FIX}\,(f) \mid f \in Z\,\}$ ist. Sei also $f \in Z$. Da $f \sqsubseteq' \bigsqcup' Z$ und f und $\bigsqcup' Z$ monoton und kettenstetig sind, folgt aus Lem. 45, dass $\mathrm{FIX}\,(f) \sqsubseteq \mathrm{FIX}\left(\bigsqcup' Z\right)$. □

Jetzt haben wir das Werkzeug, um die Wohldefiniertheit von $\mathcal{C}\,[\![\_]\!]$ zu zeigen.

Theorem 47. $\mathcal{C}\,[\![c]\!]\ f\ S$ ist wohldefiniert, monoton und kettenstetig in f und S.

Beweis. Induktion über c.

- Fall $c = \texttt{skip}$: Die Funktion $\lambda f.\ \mathcal{C}\,[\![\texttt{skip}]\!]\ f\ S$ ist die Identität, die trivialerweise monoton und kettenstetig ist. Ebenso ist $\lambda S.\ \mathcal{C}\,[\![\texttt{skip}]\!]\ f\ S$ als konstante Funktion monoton und kettenstetig.
- Fall $c = x\ \texttt{:=}\ a$: Es gilt $\mathcal{C}\,[\![x\ \texttt{:=}\ a]\!]\ f\ S$ ist konstant in S, also trivialerweise monoton und kettenstetig in S. Da $\mathcal{C}\,[\![x\ \texttt{:=}\ a]\!]\ f\ S = (\lambda f.\ f \circ (\lambda\sigma.\ \sigma[x \mapsto \mathcal{A}\,[\![a]\!]\,\sigma]))\ f$ haben wir die Monotonie und Kettenstetigkeit bereits im Beweis von Thm. 34 gezeigt.
- Fall $c = c_1\,;\ c_2$: In f: $\mathcal{C}\,[\![c_1\,;\ c_2]\!]\ f\ S = ((\lambda f.\ \mathcal{C}\,[\![c_1]\!]\ f\ S) \circ (\lambda f.\ \mathcal{C}\,[\![c_2]\!]\ f\ S))\ f$ ist die Hintereinanderausführung der Funktionen $\lambda f.\ \mathcal{C}\,[\![c_1]\!]\ f\ S$ und $\lambda f.\ \mathcal{C}\,[\![c_2]\!]\ f\ S$, die nach Induktionsannahme monoton und kettenstetig sind. Damit ist auch ihre Hintereinanderausführung monoton und kettenstetig.

  In S: Wegen $\mathcal{C}\,[\![c_1\,;\ c_2]\!]\ f\ S = (\lambda S f.\ \mathcal{C}\,[\![c_1]\!]\ f\ S)\ S\ (\mathcal{C}\,[\![c_2]\!]\ f\ S)$ folgt Monotonie und Kettenstetigkeit direkt durch Anwendung von Lem. 43, dessen Annahmen genau die Induktionsannahmen sind.
- Fall $c = \texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2$: Auch $\mathcal{C}\,[\![\texttt{if (}b\texttt{) then}\ c_1\ \texttt{else}\ c_2]\!]$ ist nur eine Kombination kettenstetiger Funktionen: $\lambda f\ g.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ f,\ g)$ ist monoton und kettenstetig in f und g, $\mathcal{C}\,[\![c_1]\!]$ und $\mathcal{C}\,[\![c_2]\!]$ sind nach Induktionsannahme monoton und kettenstetig.
- Fall $c = \texttt{while (}b\texttt{) do}\ c'$:
  Induktionsannahme: $\mathcal{C}\,[\![c']\!]$ ist monoton und kettenstetig in beiden Argumenten.

  Bezeichne mit $F(f)(S) = \lambda g.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ \mathcal{C}\,[\![c']\!]\ g\ S,\ f)$ das Funktional.
  – Wohldefiniertheit: Das Funktional $F(f)(S)$ ist für beliebige f und S monoton und kettenstetig. Wie schon bei der direkten denotationalen Semantik für While lässt sich $F(f)(S)$ als Hintereinanderausführung schreiben: $F(f)(S) = (\lambda g.\ \mathrm{IF}\,(\mathcal{B}\,[\![b]\!],\ g,\ f)) \circ (\lambda g.\ \mathcal{C}\,[\![c']\!]\ g\ S)$. In Kapitel 6.3 haben wir schon gezeigt, dass der vordere Teil monoton und kettenstetig ist, der hintere erfüllt dies laut Induktionsannahme. NB: Für diesen Schritt haben wir den Induktionsbeweis aufgezogen.
  – Monotonie und Kettenstetigkeit: Da das Funktional F monoton und kettenstetig ist, ist FIX auch monoton und kettenstetig (Lem. 45 und 46). Damit haben wir auch hier wieder eine Kombination monotoner und kettenstetiger Funktionen, d.h., $\mathcal{C}\,[\![\texttt{while (}b\texttt{) do}\ c']\!]$ selbst ist auch monoton und kettenstetig.
- Fall $c = \texttt{raise}\ X$: Folgt direkt durch ausrechnen, da Y nicht-leer ist:

$$\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ f_1\ S_1 = S_1(X) \sqsubseteq S_2(X) = \mathcal{C}\,[\![\texttt{raise}\ X]\!]\ f_2\ S_2 \qquad \text{für } f_1 \sqsubseteq f_2 \text{ und } S_1 \sqsubseteq' S_2$$

$$\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ \left(\bigsqcup Y\right)\ S = S(X) = \bigsqcup \{\,S(X) \mid f \in Y\,\} = \bigsqcup \{\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ f\ S \mid f \in Y\,\}$$

$$\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ f\ \left(\bigsqcup' Z\right) = \left(\bigsqcup' Z\right)(X) = \bigsqcup \{\,S(X) \mid S \in Z\,\} = \bigsqcup \{\,\mathcal{C}\,[\![\texttt{raise}\ X]\!]\ f\ S \mid S \in Z\,\}$$

- Fall $c = \texttt{try}\ c_1\ \texttt{catch}\ X\ c_2$: Induktionsannahmen: $\mathcal{C}\,[\![c_1]\!]$ und $\mathcal{C}\,[\![c_2]\!]$ sind monoton und kettenstetig in beiden Argumenten.

  Da $\mathcal{C}\,[\![\texttt{try}\ c_1\ \texttt{catch}\ X\ c_2]\!]\ f\ S = \mathcal{C}\,[\![c_1]\!]\ f\ ((\lambda f.\ S[X \mapsto f])(\mathcal{C}\,[\![c_2]\!]\ f\ S))$, ist auch
  $\mathcal{C}\,[\![\texttt{try}\ c_1\ \texttt{catch}\ X\ c_2]\!]\ f\ S$ nur eine Kombination kettenstetiger Funktionen (Induktionsannahme und Lem. 42), und damit selbst auch monoton und kettenstetig in f.

  Für S teilt man $\mathcal{C}\,[\![\texttt{try}\ c_1\ \texttt{catch}\ X\ c_2]\!]\ f\ S$ in $(\mathcal{C}\,[\![c_1]\!]\ f \circ h)(S)$ auf, wobei
  $h(S) = (\lambda S\ s.\ S[X \mapsto s])(S)(\mathcal{C}\,[\![c_2]\!]\ s\ S)$. h entspricht dann wieder der Form von Lem. 43 und damit ist $\mathcal{C}\,[\![\texttt{try}\ c_1\ \texttt{catch}\ X\ c_2]\!]\ f\ S$ auch monoton und kettenstetig in S. □