

Beispiel: Summen-Programm (Monotonie)

$S_{ds} \llbracket y := 0 ; \text{while } \neg(x = 0) \text{ do } (y := y + x; x := x - 1) \rrbracket s$
 $= (\text{FIX } F)(s[y \mapsto 0]),$

wobei

$$F(g) \ s = \begin{cases} g(s[y \mapsto (s \ y + s \ x)] [x \mapsto (s \ x - 1)]), & \text{falls } s \ x \neq 0 \\ s, & \text{falls } s \ x = 0 \end{cases}$$

F ist monoton!

61

Stetige Funktionen

geben $(D, \sqsubseteq), (D', \sqsubseteq')$ ccpo's und $f: D \rightarrow D'$ eine (totale) Funktion. Dann heit f *stetig*, falls

- f monoton ist, und
- $\sqcup' \{f \ d \mid d \in Y\} = f(\sqcup Y)$ fr alle nicht-leeren Ketten Y von D .

bung: (Stetigkeit impliziert Monotonie)

geben $(D, \sqsubseteq), (D', \sqsubseteq')$ ccpo's und $f: D \rightarrow D'$ eine (totale) Funktion mit $\sqcup' \{f \ d \mid d \in Y\} = f(\sqcup Y)$ fr alle nicht-leeren Ketten Y von D . Dann ist f auch monoton.

Beweis(idee): Betrachte fr beliebige $d_1, d_2 \in D$ mit $d_1 \sqsubseteq d_2$ die Ketten $Y = \{d_1, d_2\}$ in D bzw. $f \ Y = \{f \ d_1, f \ d_2\}$ in D' , und leite daraus $f \ d_1 \sqsubseteq' f \ d_2$ ab.

63

Monotonie nicht ausreichend!

fr Gleichheit in: $\sqcup' \{f \ d \mid d \in Y\} \sqsubseteq' f(\sqcup Y)$
 (wobei $(D, \sqsubseteq), (D', \sqsubseteq')$ ccpo's und $f: D \rightarrow D'$ monoton):

Beispiel: Sei $f: \mathcal{P}(\mathbb{N} \cup \{a\}) \rightarrow \mathcal{P}(\mathbb{N} \cup \{a\})$ gegeben durch

$$f \ X = \begin{cases} X, & \text{falls } X \text{ endlich} \\ X \cup \{a\}, & \text{falls } X \text{ unendlich} \end{cases}$$

Dann ist f (offensichtlich) monoton.

Aber $\sqcup' \{f \ d \mid d \in Y\} = f(\sqcup Y)$ gilt nicht immer, d.h. nicht fr alle Ketten $Y \subseteq \mathbb{N} \cup \{a\}$: Betrachte $Y = \{\{0, 1, \dots, n\} \mid n \geq 0\}$:

- zwar: $\sqcup \{f \ X \mid X \in Y\} = \sqcup Y = \mathbb{N}$
- aber: $f(\sqcup Y) = f \ \mathbb{N} = \mathbb{N} \cup \{a\}$
- also: $\sqcup \{f \ X \mid X \in Y\} \neq f(\sqcup Y)$!

62

Beispiel: Summen-Programm

$S_{ds} \llbracket y := 0 ; \text{while } \neg(x = 0) \text{ do } (y := y + x; x := x - 1) \rrbracket s$
 $= (\text{FIX } F)(s[y \mapsto 0]),$

wobei

$$(F \ g) \ s = \begin{cases} g(s[y \mapsto (s \ y + s \ x)] [x \mapsto (s \ x - 1)]), & \text{falls } s \ x \neq 0 \\ s, & \text{falls } s \ x = 0 \end{cases}$$

F ist stetig!

64

Fixpunkt-Satz

Satz (Fixpunkttheorem): Sei $f : D \rightarrow D$ eine stetige Funktion auf der ccpo (D, \sqsubseteq) mit kleinstem Element \perp . Dann ist

- $\text{FIX } f = \sqcup \{f^n \perp \mid n \geq 0\}$ wohldefiniert und ein Element aus D , und

- FIX ist der kleinste Fixpunkt von f .

Notation:

- $f^0 = \text{id}$

- $f^{n+1} = f \circ f^n$, für $n \geq 0$

Beweis(struktur): Zeige

- 1) $\text{FIX } f$ ist wohldefiniert.
- 2) $\text{FIX } f$ ist Fixpunkt von f .
- 3) $\text{FIX } f$ ist der kleinste Fixpunkt von f .

65

Beispiel: Summen-Programm (konstruktive Fixpunkt-Berechnung)

$$F^0 \perp) s = \text{undef}$$

$$F^1 \perp) s = \begin{cases} \text{undef,} & \text{falls } s \ x \neq 0 \\ s, & \text{falls } s \ x = 0 \end{cases}$$

$$F^2 \perp) s = \begin{cases} \text{undef,} & \text{falls } s \ x \neq 0 \text{ und } s \ x \neq 1 \\ s[y \mapsto (s \ y + s \ x)][x \mapsto (s \ x - 1)], & \text{falls } s \ x = 1 \\ s, & \text{falls } s \ x = 0 \end{cases}$$

$$F^n \perp) s = \begin{cases} \text{undef,} & \text{falls } s \ x < 0 \text{ oder } s \ x > n \\ s[y \mapsto (s \ y + s \ x + (s \ x - 1) + \dots + 1)][x \mapsto 0], & \text{falls } 0 \leq s \ x \leq n \end{cases}$$

$$\text{FIX } F) s = \begin{cases} \text{undef,} & \text{falls } s \ x < 0 \\ s[y \mapsto (s \ y + \sum_{k=1}^n k)][x \mapsto 0], & \text{falls } n = s \ x \geq 0 \end{cases}$$

67

Beispiel: Summen-Programm (Voraussetzungen für Fixpunktsatz)

$$\begin{aligned} \mathcal{S}_{ds} \llbracket y := 0 ; \text{while } \neg(x=0) \text{ do } (y := y + x; x := x - 1) \rrbracket s \\ = (\text{FIX } F)(s[y \mapsto 0]), \end{aligned}$$

wobei

$$(F \ g) \ s = \begin{cases} g(s[y \mapsto (s \ y + s \ x)][x \mapsto (s \ x - 1)]), & \text{falls } s \ x \neq 0 \\ s, & \text{falls } s \ x = 0 \end{cases}$$

Zur (konstruktiven) Berechnung der denotationalen Semantik mittels Fixpunktsatz zu zeigen:

- (1) $(\text{State} \hookrightarrow \text{State}, \sqsubseteq)$ ist eine ccpo (\rightarrow siehe später).
- (2) F ist eine stetige Funktion (\rightarrow siehe später, für beliebige Programme).

66

Denotationale Semantik: Definition

$$\mathcal{S}_{ds} : \text{Prog} \rightarrow (\text{State} \hookrightarrow \text{State})$$

$$[\text{assds}] \quad \mathcal{S}_{ds} \llbracket x := a \rrbracket s = s[x \mapsto \mathcal{A} \llbracket a \rrbracket s]$$

$$[\text{skipds}] \quad \mathcal{S}_{ds} \llbracket \text{skip} \rrbracket = \text{id}$$

$$[\text{compds}] \quad \mathcal{S}_{ds} \llbracket P_1; P_2 \rrbracket = \mathcal{S}_{ds} \llbracket P_2 \rrbracket \circ \mathcal{S}_{ds} \llbracket P_1 \rrbracket$$

$$[\text{compds}] \quad \mathcal{S}_{ds} \llbracket \text{if } b \text{ then } P_1 \text{ else } P_2 \rrbracket = \text{cond}(\mathcal{B} \llbracket b \rrbracket, \mathcal{S}_{ds} \llbracket P_1 \rrbracket, \mathcal{S}_{ds} \llbracket P_2 \rrbracket)$$

$$[\text{whileds}] \quad \mathcal{S}_{ds} \llbracket \text{while } b \text{ do } P \rrbracket = \text{FIX } F$$

mit

$$F : (\text{State} \hookrightarrow \text{State}) \rightarrow (\text{State} \hookrightarrow \text{State})$$

$$F \ g = \text{cond}(\mathcal{B} \llbracket b \rrbracket, g \circ \mathcal{S}_{ds} \llbracket P \rrbracket, \text{id})$$

68

Denotationale Semantik: Haupterg. (1)

- **Satz (Existenz und Berechnung kleinster Fixpunkte):**

Sei $f : D \rightarrow D$ eine stetige Funktion auf der cpo (D, \sqsubseteq) mit kleinstem Element \perp . Dann definiert

$$\text{FIX } f = \bigsqcup \{f^n \perp \mid n \geq 0\}$$

ein Element von D , und zwar den kleinsten Fixpunkt von f .

- **Lemma (Kettenvollständigkeit der Definiertheitsordnung):**

$(\text{State} \rightarrow \text{State}, \sqsubseteq)$ ist eine kettenvollständige Partialordnung (d.h., eine cpo). Die kleinste obere Schranke $\bigsqcup Y$ einer Kette Y in $(\text{State} \rightarrow \text{State}, \sqsubseteq)$ ist gegeben durch

$$\left(\bigsqcup Y\right) s = \begin{cases} g s, & \text{falls } g s \neq \text{undef} \\ \text{undef}, & \text{sonst} \end{cases}$$

- **Satz (Wohldefiniertheit der denotationalen Semantik S_{ds}):**

Die semantischen Gleichungen für S_{ds} definieren eine totale Funktion $S_{ds} : \text{Prog} \rightarrow (\text{State} \hookrightarrow \text{State})$.

69

Denotationale Semantik: Haupterg. (3)

Satz (Wohldefiniertheit der denotationalen Semantik S_{ds}):

Die semantischen Gleichungen für S_{ds} definieren eine totale Funktion $S_{ds} : \text{Prog} \rightarrow (\text{State} \hookrightarrow \text{State})$.

Beweis(struktur): Prinzip: Strukturelle Induktion über den Aufbau von Programmen.

- **IA:** Fall (1a), $x := a$ und Fall (1b), **skip**: Klar.
- **IS:** Im Ind.schluß sind 3 Fälle zu betrachten, Komposition, bedingte Anweisung und **while**-Schleife. In allen 3 Fällen läßt sich die Behauptung mit der jeweiligen Induktionshypothese zeigen (im “**while**-Fall” werden Hilfslemmata benötigt!).

71

Denotationale Semantik: Haupterg. (2)

Lemma (Kettenvollständigkeit der Definiertheitsordnung):

$(\text{State} \rightarrow \text{State}, \sqsubseteq)$ ist eine kettenvollständige Partialordnung (d.h., eine cpo). Die kleinste obere Schranke $\bigsqcup Y$ einer Kette Y in $(\text{State} \rightarrow \text{State}, \sqsubseteq)$ ist gegeben durch

$$\left(\bigsqcup Y\right) s = \begin{cases} g s, & \text{falls } g s \neq \text{undef} \\ \text{undef}, & \text{sonst} \end{cases}$$

Beweis(struktur): Zeige:

- $\bigcup \{graph(g) \mid g \in Y\}$ ist der Graph einer partiellen Funktion g_0 in $\text{State} \rightarrow \text{State}$.^a

- g_0 ist obere Schranke von Y .

- g_0 ist kleinste obere Schranke von Y .

^aFür eine (partielle) Funktion $g : X \hookrightarrow Y$ ist $graph(g)$ definiert durch $graph(g) = \{(x, y) \in X \times Y \mid g x \text{ definiert und } g x = y\}$.

70

Denotationale Semantik: Hilfslemmata

- **Lemma (Konditional stetig im 2. Argument):**

Seien $g_0 : \text{State} \hookrightarrow \text{State}$, $p : \text{State} \rightarrow \mathbf{T}$ gegeben und sei

$$F g = \text{cond}(p, g, g_0) .$$

Dann ist F stetig.

Beweis: Übung.

- **Lemma (Komposition stetig im 1. Argument):**

Sei $g_0 : \text{State} \hookrightarrow \text{State}$ gegeben und sei

$$F g = g \circ g_0 .$$

Dann ist F stetig.

Beweis: Übung.

- **Lemma (Komposition erhält Stetigkeit):**

Falls $f : D \rightarrow D'$ und $f' : D' \rightarrow D''$ stetige Funktionen sind, so ist auch $f' \circ f$ eine stetige Funktion.

Beweis: Übung.

72

Äquivalenz von DS und SoS (sowie NS) (1)

zisher:

- natürliche Semantik (NS), strukturelle operationale Semantik (SoS), denotationale Semantik (DS)
- bereits gezeigt: NS und SoS sind äquivalent
- Zusammenhang NS/SoS – DS?

Satz ($\mathcal{S}_{sos} = \mathcal{S}_{ds}$):

für jedes Programm $P \in \mathbf{Prog}$ gilt

$$\mathcal{S}_{sos} \llbracket P \rrbracket = \mathcal{S}_{ds} \llbracket P \rrbracket$$

wobei

$$\mathcal{S}_{sos} \llbracket P \rrbracket s = \begin{cases} s', & \text{falls } \langle P, s \rangle \Rightarrow^* s' \\ \text{undefiniert,} & \text{sonst} \end{cases}$$

73

Den. Semantik: Spracherweiterungen (hier nicht!)

- Deklarationen (Variablen, Prozeduren)
- Umgebungsblöcke mit lokalen Deklarationen
- Prozeduraufrufe, rekursive Prozeduraufrufe
- Typisierung / Typkonzepte
- (*) Sprünge, Ausnahmen (exceptions)

Konsequenzen für Semantik(modellierung/definition)?

- Gültigkeitsbereiche von Deklarationen (statisch / dynamisch)
- Programmuzustände: Zuordnungen der Form State genügen nicht mehr!
- (*): Denotationale Semantik mit *continuations* geeignet. Grundidee: Die Fortsetzung (*continuation*) eines Programmteiles ist das Ergebnis der Ausführung des Restprogramms.

75

Äquivalenz von DS und SoS (sowie NS) (2)

Zwei Richtungen für $\mathcal{S}_{sos} \llbracket P \rrbracket = \mathcal{S}_{ds} \llbracket P \rrbracket$:

Lemma 1:

Für jedes Programm $P \in \mathbf{Prog}$ gilt: $\mathcal{S}_{sos} \llbracket P \rrbracket \sqsubseteq \mathcal{S}_{ds} \llbracket P \rrbracket$.

Beweis(struktur):

- Für 1-schrittige \Rightarrow -Ableitungen strukturelle Induktion über den SoS-Ableitungsbaum, sodann
- für mehrschrittige \Rightarrow -Ableitungen eine Induktion über die Ableitungslänge.

Lemma 2:

Für jedes Programm $P \in \mathbf{Prog}$ gilt: $\mathcal{S}_{ds} \llbracket P \rrbracket \sqsubseteq \mathcal{S}_{sos} \llbracket P \rrbracket$.

Beweis(struktur): Strukturelle Induktion über den Aufbau von Programmen.

74

Semantik von Programmiersprachen (1)

Programmierparadigmen:

- *imperativ*:
im wesentlichen gut erforscht und verstanden
- *funktional*:
im wesentlichen gut erforscht und verstanden
- *nebenläufig (concurrent)*:
relativ wenig (vor allem theoretisch) erforscht
- *objektorientiert*:
kaum erforscht
- *logisch*:
im wesentlichen gut erforscht und verstanden
- ...

76

Semantik von Programmiersprachen (2)

Kombinierte Programmierparadigmen:

- *funktional + imperativ*:
im wesentlichen gut erforscht und verstanden
- *funktional + nebenläufig*:
kaum erforscht
- *funktional + objektorientiert*:
kaum erforscht
- *funktional + logisch*:
einigermaßen gut erforscht und verstanden
- ...

77

Zusammenfassung

Hier vorgestellte Grundkonzepte:

- Natürliche Semantik (operationale Gesamtschritt-Semantik)
- Strukturelle operationale Semantik (operationale Einzelschritt-Semantik)
- Denotationale Semantik (*direct style* Version, abstrakter Ansatz; Semantik als mathematische Funktion)

Wichtig:

- Zugrundeliegende Überlegungen und Intentionen
- Entsprechende induktive Beweisprinzipien, Zusammenhänge
- Wie funktioniert konkrete Semantik-Berechnung?

Anwendung / Vertiefung:

- Bei jeder Programmiersprache (im Prinzip)
- Speziallehveranstaltungen (später)

78