
Semantikdefinitionsstile (1)

Es gibt unterschiedliche Stile, die Semantik einer Programmiersprache festzulegen. Sie richten sich an unterschiedliche Adressaten und deren spezifische Sicht auf die Semantik...

Insbesondere unterscheiden wir den...

- *denotationellen*
- *operationellen*
- *axiomatischen*

Stil.

Semantikdefinitionsstile (2)

- *Sprachentwicklersicht*
 - Denotationelle Semantik
- *Sprach- und Anwendungsimplementierersicht*
 - Operationelle Semantik
 - * Strukturell operationelle Semantik (small steps semantics)
 - * Natürliche Semantik (big steps semantics)
- *Programmierer- und Verifizierersicht*
 - Axiomatische Semantik

Strukturell operationelle Semantik (1)

...noch einmal wiederholt für das Beispiel von WHILE:

$$[\text{skip}_{\text{sos}}] \frac{}{\langle \text{skip}, \sigma \rangle \Rightarrow \sigma}$$

$$[\text{abort}_{\text{sos}}] \frac{}{\langle \text{abort}, \sigma \rangle \Rightarrow \text{error}}$$

$$[\text{ass}_{\text{sos}}] \frac{}{\langle x := t, \sigma \rangle \Rightarrow \sigma[\llbracket t \rrbracket_A(\sigma) / x]}$$

$$[\text{comp}_{\text{sos}}^1] \frac{\langle \pi_1, \sigma \rangle \Rightarrow \langle \pi_1', \sigma' \rangle}{\langle \pi_1; \pi_2, \sigma \rangle \Rightarrow \langle \pi_1; \pi_2, \sigma' \rangle}$$

$$[\text{comp}_{\text{sos}}^2] \frac{\langle \pi_1, \sigma \rangle \Rightarrow \sigma'}{\langle \pi_1; \pi_2, \sigma \rangle \Rightarrow \langle \pi_2, \sigma' \rangle}$$

Strukturell operationelle Semantik (2)

$$[\text{if}_{\text{sos}}^{\text{tt}}] \frac{}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \Rightarrow \langle \pi_1, \sigma \rangle} \quad \llbracket b \rrbracket_B(\sigma) = \text{tt}$$

$$[\text{if}_{\text{sos}}^{\text{ff}}] \frac{}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \Rightarrow \langle \pi_2, \sigma \rangle} \quad \llbracket b \rrbracket_B(\sigma) = \text{ff}$$

$$[\text{while}_{\text{sos}}] \frac{}{\langle \text{while } b \text{ do } \pi \text{ od}, \sigma \rangle \Rightarrow \langle \text{if } b \text{ then } \pi; \text{ while } b \text{ do } \pi \text{ od else skip fi}, \sigma \rangle}$$

Strukturell operationelle Semantik (3)

Der Fokus liegt auf...

- *individuellen Schritten* einer Berechnungsfolge, d.h. auf der Ausführung von Zuweisungen und Tests

Intuitive Bedeutung der Transitionsrelation...

$$\langle \pi, \sigma \rangle \Rightarrow \gamma$$

...mit γ von der Form $\langle \pi', \sigma' \rangle$ oder σ' oder *error* beschreibt den *ersten* Schritt der Berechnungsfolge von π angesetzt auf σ . Folgende Übergänge sind möglich:

- γ von der Form $\langle \pi', \sigma' \rangle$:
Abarbeitung von π nicht vollständig; das Restprogramm π' ist auf σ' anzusetzen
- γ von der Form σ' :
Abarbeitung von π vollständig; π angesetzt auf σ terminiert in einem Schritt in σ'
- γ von der Form *error*:
Abarbeitung von π terminiert irregulär

Natürliche Semantik (1)

...ebenfalls für das Beispiel von WHILE:

$$[\text{skip}_{ns}] \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}$$

$$[\text{abort}_{ns}] \frac{}{\langle \text{abort}, \sigma \rangle \rightarrow \text{error}}$$

$$[\text{ass}_{ns}] \frac{}{\langle x := t, \sigma \rangle \rightarrow \sigma[\llbracket t \rrbracket_A(\sigma) / x]}$$

$$[\text{comp}_{ns}] \frac{\langle \pi_1, \sigma \rangle \rightarrow \sigma', \langle \pi_2, \sigma' \rangle \rightarrow \sigma''}{\langle \pi_1; \pi_2, \sigma \rangle \rightarrow \sigma''}$$

Natürliche Semantik (2)

$$[\text{if}_{ns}^{tt}] \frac{\langle \pi_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \rightarrow \sigma'} \quad \llbracket b \rrbracket_B(\sigma) = \text{tt}$$

$$[\text{if}_{ns}^{ff}] \frac{\langle \pi_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi}, \sigma \rangle \rightarrow \sigma'} \quad \llbracket b \rrbracket_B(\sigma) = \text{ff}$$

$$[\text{while}_{ns}^{tt}] \frac{\langle \pi, \sigma \rangle \rightarrow \sigma', \langle \text{while } b \text{ do } \pi \text{ od}, \sigma' \rangle \rightarrow \sigma''}{\langle \text{while } b \text{ do } \pi \text{ od}, \sigma \rangle \rightarrow \sigma''} \quad \llbracket b \rrbracket_B(\sigma) = \text{tt}$$

$$[\text{while}_{ns}^{ff}] \frac{}{\langle \text{while } b \text{ do } \pi \text{ od}, \sigma \rangle \rightarrow \sigma} \quad \llbracket b \rrbracket_B(\sigma) = \text{ff}$$

Natürliche Semantik (3)

Der Fokus liegt auf...

- Zusammenhang von *initialem* und *finalelem* Zustand einer Berechnungsfolge

Intuitive Bedeutung von...

$$\langle \pi, \sigma \rangle \rightarrow \gamma$$

...mit γ von der Form σ' oder *error* ist: π angesetzt auf initialen Zustand σ terminiert schließlich im finalen Zustand σ' bzw. terminiert irregulär.

Determinismus der NS-Regeln

Lemma 2.1

$\forall \pi \in \mathbf{Prg}, \sigma \in \Sigma, \gamma, \gamma' \in \Gamma. \langle \pi, \sigma \rangle \rightarrow \gamma \wedge \langle \pi, \sigma \rangle \rightarrow \gamma' \Rightarrow \gamma = \gamma'$

Korollar 2.2

Die von den NS-Regeln für eine Konfiguration induzierte finale Konfiguration ist (sofern definiert) eindeutig bestimmt, d.h. *deterministisch*.

Salopper, wenn auch weniger präzise:

Die (N-) Semantik von WHILE ist deterministisch!

Das Semantikfunktional $\llbracket \cdot \rrbracket_{ns}$

Korollar 2.2 erlaubt uns festzulegen:

- Die natürliche Semantik von WHILE ist gegeben durch das Funktional

$$\llbracket \cdot \rrbracket_{ns} : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

welches definiert wird durch:

$$\forall \pi \in \mathbf{Prg}, \sigma \in \Sigma. \llbracket \pi \rrbracket_{ns}(\sigma) =_{df} \begin{cases} \sigma' & \text{falls } \langle \pi, \sigma \rangle \rightarrow \sigma' \\ error & \text{falls } \langle \pi, \sigma \rangle \rightarrow error \\ undef & \text{sonst} \end{cases}$$

Variante induktiver Beweisführung

Induktion über die Form von Ableitungsbäumen:

- *Induktionsanfang*
 - Beweise, dass A für die Axiome des Transitionssystems gilt (und somit für alle nichtzusammengesetzte Ableitungsbäume).
- *Induktionsschritt*
 - Beweise für jede echte Regel des Transitionssystems unter der Annahme, dass A für jede Prämisse dieser Regel gilt (*Induktionshypothese!*), A auch für die Konklusion dieser Regel gilt, sofern die (ggf. vorhandenen) Randbedingungen der Regel erfüllt sind.

Anwendung

- Induktive Beweisführung über die Form von Ableitungsbäumen ist typisch zum Nachweis von Aussagen über Eigenschaften natürlicher Semantik.

Ein Beispiel dafür ist der Beweis von **Lemma 2.1!**

Denotationelle Semantik (1)

...auch für das Beispiel von WHILE:

$$\llbracket \text{skip} \rrbracket_{ds} = Id$$

$$\llbracket \text{abort} \rrbracket_{ds} = Error$$

$$\llbracket x := t \rrbracket_{ds}(\sigma) = \sigma[\llbracket t \rrbracket_A(\sigma)/x]$$

$$\llbracket \pi_1; \pi_2 \rrbracket_{ds} = \llbracket \pi_2 \rrbracket_{ds} \circ \llbracket \pi_1 \rrbracket_{ds}$$

$$\llbracket \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \rrbracket_{ds} = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, \llbracket \pi_1 \rrbracket_{ds}, \llbracket \pi_2 \rrbracket_{ds})$$

$$\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds} = FIX F$$

$$\text{where } F g = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, g \circ \llbracket \pi \rrbracket_{ds}, Id)$$

Denotationelle Semantik (2)

Es bezeichnen:

- $Id : \Sigma_{\varepsilon} \rightarrow \Sigma_{\varepsilon}$ die identische Zustandstransformation:

$$\forall \sigma \in \Sigma_{\varepsilon}. Id(\sigma) =_{df} \sigma$$

- $Error : \Sigma_{\varepsilon} \rightarrow \Sigma_{\varepsilon}$ die konstante Zustandstransformation mit:

$$\forall \sigma \in \Sigma_{\varepsilon}. Error(\sigma) =_{df} error$$

Denotationelle Semantik (3)

Zur Hilfsfunktion *cond*...

Funktionalität...

$$\text{cond} : (\Sigma \rightarrow \mathcal{B}) \times (\Sigma \rightarrow \Sigma) \times (\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)$$

Definiert durch...

$$\text{cond}(p, g_1, g_2) \sigma =_{df} \begin{cases} g_1 \sigma & \text{falls } p \sigma = \text{tt} \\ g_2 \sigma & \text{falls } p \sigma = \text{ff} \end{cases}$$

Zu den Argumenten und zum Resultat von *cond*...

- 1. Argument: Prädikat (in unserem Szenario total definiert; siehe Vorlesungsteil 1)
- 2.&3. Argument: Je eine partiell definierte Zustandstransformation
- Resultat: Wieder eine partiell definierte Zustandstransformation

Denotationelle Semantik (4)

Zur Hilfsfunktion *FIX*...

Funktionalität...

$$FIX : ((\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)) \rightarrow (\Sigma \rightarrow \Sigma)$$

Definiert durch...

$$F g = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, g \circ \llbracket \pi \rrbracket_{ds}, Id)$$

Daraus ergibt sich...

- *FIX* ist ein Funktional ("Zustandstransformationsfunktional")
- Die denotationelle Semantik der while-Schleife ist ein Fixpunkt des Funktionals *F* (und zwar der kleinste!)

Mehr Details zu FIX und Co. später!

Denotationelle Semantik (5)

- *Operationelle* Semantik
...der Fokus liegt darauf, *wie* ein Programm ausgeführt wird
- *Denotationelle* Semantik
...der Fokus liegt auf dem *Effekt*, den die Ausführung eines Programms hat: Für jedes *syntaktische* Konstrukt gibt es eine *semantische* Funktion, die ersterem ein *mathematisches Objekt* zuweist, i.a. eine Funktion, die den Effekt der Ausführung des Konstrukts beschreibt (jedoch nicht, wie dieser Effekt erreicht wird).

Denotationelle Semantik (6)

Zentral für denotationelle Semantiken: **Kompositionalität!**

Intuitiv:

- Für jedes Element der elementaren syntaktischen Konstrukte/Kategorien gibt es eine zugehörige semantische Funktion
- Für jedes Element eines zusammengesetzten syntaktischen Konstrukts/Kategorie gibt es eine semantische Funktion, die über die semantischen Funktionen der Komponenten des zusammengesetzten Konstrukts definiert ist.

Denotationelle Semantik (7)

Lemma 2.2

Für alle $\pi \in \mathbf{Prg}$ ist durch die Gleichungen von Folie "Denotationelle Semantik (1)" eine (partielle) Funktion $\llbracket \pi \rrbracket_{ds}$ definiert, die denotationelle Semantik von π .

Hauptergebnisse

Theorem

$$\forall \pi \in \mathbf{Prg}. \llbracket \pi \rrbracket_{sos} = \llbracket \pi \rrbracket_{ns} = \llbracket \pi \rrbracket_{ds}$$

Die Äquivalenz der strukturell operationellen, natürlichen und denotationellen Semantik von WHILE legt es nahe, den semantikangebenden Index in der Folge fortzulassen und vereinfachend von $\llbracket \cdot \rrbracket$ als der Semantik der Sprache WHILE zu sprechen:

$$\llbracket \cdot \rrbracket : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

definiert durch

$$\llbracket \cdot \rrbracket =_{df} \llbracket \cdot \rrbracket_{sos}$$

WHILE – Denotationelle Semantik (1)

- **Prg** ...bezeichne die Menge aller Programme der Sprache **WHILE**

Denotationelle Semantik

$$\llbracket \cdot \rrbracket_{ds} : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

Somit...

- Die *denotationelle Semantik* eines **WHILE**-Programms ist eine (partiell definierte) *Zustandstransformation*, wobei die Menge der *Zustände* gegeben ist durch

$$\Sigma =_{df} \{ \sigma \mid \sigma : V \rightarrow D \}$$

Beachte...

- Auch die operationelle (die strukturell operationelle wie auch die natürliche) Semantik eines **WHILE**-Programms ist eine (partiell definierte) *Zustandstransformation* auf Σ , nicht aber die axiomatische Semantik.

WHILE – Denotationelle Semantik (2)

Erinnerung:

$$\llbracket skip \rrbracket_{ds} = Id$$

$$\llbracket abort \rrbracket_{ds} = Error$$

$$\llbracket x := t \rrbracket_{ds}(\sigma) = \sigma[\llbracket t \rrbracket_A(\sigma)/x]$$

$$\llbracket \pi_1; \pi_2 \rrbracket_{ds} = \llbracket \pi_2 \rrbracket_{ds} \circ \llbracket \pi_1 \rrbracket_{ds}$$

$$\llbracket \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \rrbracket_{ds} = cond(\llbracket b \rrbracket_B, \llbracket \pi_1 \rrbracket_{ds}, \llbracket \pi_2 \rrbracket_{ds})$$

$$\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds} = FIX F$$

$$\text{where } F g = cond(\llbracket b \rrbracket_B, g \circ \llbracket \pi \rrbracket_{ds}, Id)$$

WHILE – Denotationelle Semantik (3)

Noch offen...

- Die Bedeutung von...

– *cond* und

– *FIX F*

Diese Bedeutung wollen wir in der Folge aufklären...

Zur Bedeutung von cond

Hilfsfunktion *cond*...

Funktionalität...

$$cond : (\Sigma \rightarrow \mathbf{B}) \times (\Sigma \rightarrow \Sigma) \times (\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)$$

Definiert durch...

$$cond(p, g_1, g_2) \sigma =_{df} \begin{cases} g_1 \sigma & \text{if } p \sigma = \text{tt} \\ g_2 \sigma & \text{if } p \sigma = \text{ff} \end{cases}$$

Zu den Argumenten und zum Resultat von *cond*...

- 1. Argument: Prädikat (in unserem Szenario total definiert; siehe Vorlesungsteil 1)
- 2.&3. Argument: Je eine partiell definierte Zustandstransformation
- Resultat: Wieder eine partiell definierte Zustandstransformation

Damit erhalten wir

...für die Bedeutung der Fallunterscheidung

$$\begin{aligned} & \llbracket \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \rrbracket_{ds} \sigma \\ &= \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, \llbracket \pi_1 \rrbracket_{ds}, \llbracket \pi_2 \rrbracket_{ds}) \sigma \\ &= \begin{cases} \sigma' & \text{falls } (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{tt} \wedge \llbracket \pi_1 \rrbracket_{ds} \sigma = \sigma') \\ & \vee (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{ff} \wedge \llbracket \pi_2 \rrbracket_{ds} \sigma = \sigma') \\ \text{error} & \text{falls } (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{tt} \wedge \llbracket \pi_1 \rrbracket_{ds} \sigma = \text{error}) \\ & \vee (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{ff} \wedge \llbracket \pi_2 \rrbracket_{ds} \sigma = \text{error}) \\ \text{undef} & \text{falls } (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{tt} \wedge \llbracket \pi_1 \rrbracket_{ds} \sigma = \text{undef}) \\ & \vee (\llbracket b \rrbracket_{\mathcal{B}} \sigma = \text{ff} \wedge \llbracket \pi_2 \rrbracket_{ds} \sigma = \text{undef}) \end{cases} \end{aligned}$$

Erinnerung:

- $\llbracket b \rrbracket_{\mathcal{B}}$ ist in unserem Szenario total definiert; $\llbracket b \rrbracket_{\mathcal{B}} \sigma$ ist daher stets von *undef* verschieden.

Zur Bedeutung von FIX F

Funktionalität...

$$FIX : ((\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)) \rightarrow (\Sigma \rightarrow \Sigma)$$

Definiert durch...

$$F g = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, g \circ \llbracket \pi \rrbracket_{ds}, Id)$$

Daraus ergibt sich...

- *FIX* ist ein Funktional ("Zustandstransformationsfunktional")
- Die denotationelle Semantik der while-Schleife ist ein Fixpunkt des Funktionals *F* (und zwar der kleinste!)

Schrittweise zur denotationellen Semantik der while-Schleife

Dazu folgende Beobachtung...

- *while b do π od* muss dieselbe Bedeutung haben wie...
if b then (π; while b do π od) else skip fi

Daraus folgt...

- $\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds} = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, \llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds} \circ \llbracket \pi \rrbracket_{ds}, Id)$

Und daraus schließlich...

- $\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds}$ muss Fixpunkt des Funktionals *F* sein, dass definiert ist durch

$$F g = \text{cond}(\llbracket b \rrbracket_{\mathcal{B}}, g \circ \llbracket \pi \rrbracket_{ds}, Id)$$

Oder anders ausgedrückt, es muss gelten:

$$\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds} = F(\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds})$$

...was uns wie gewünscht zu einer *kompositionellen* Definition von $\llbracket \text{while } b \text{ do } \pi \text{ od} \rrbracket_{ds}$ und damit von $\llbracket \rrbracket_{ds}$ insgesamt führen wird.

Etwas formaler: Unser Arbeitsplan

Erforderlich...

- Einige Resultate aus der *Fixpunkttheorie*

Beim nächsten Mal wird nachgeholt...

- Der mathematische Hintergrund (Ordnungen, CPOs, Stetigkeit von Funktionen) und die benötigten Resultate (Fixpunktsatz)
~> ...siehe Vorlesungsteil 6 vom 28.11.2006

Somit bleibt an dieser Stelle zu tun...

- Nachzuweisen, dass diese Resultate auf unsere Situation anwendbar sind.

Folgende drei Argumente...

...werden dafür entscheidend sein

1. $[\Sigma \rightarrow \Sigma]$ kann vollständig partiell geordnet werden.
2. F im Anwendungskontext ist stetig
3. Fixpunktbildung im Anwendungskontext wird ausschließlich auf stetige Funktionen angewendet.

Insgesamt ergibt sich dann daraus die Wohldefiniertheit von

$$\llbracket \cdot \rrbracket_{ds} : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

Ordnung auf Zustandstransformationen

Bezeichne...

- $[\Sigma \rightarrow \Sigma]$ die Menge der partiell definierten Zustandstransformationen.

Wir definieren...

$$g_1 \sqsubseteq g_2 \iff \forall \sigma \in \Sigma. g_1 \sigma \text{ definiert} = \sigma' \Rightarrow g_2 \sigma \text{ definiert} = \sigma' \\ \text{mit } g_1, g_2 \in [\Sigma \rightarrow \Sigma_\varepsilon]$$

Lemma 1

1. $([\Sigma \rightarrow \Sigma], \sqsubseteq)$ ist eine partielle Ordnung.
2. Die *total undefinierte* (d.h. nirgends definierte) Funktion $\perp : \Sigma \rightarrow \Sigma$ mit $\perp \sigma = \text{undef}$ für alle $\sigma \in \Sigma$ ist *kleinstes Element* in $([\Sigma \rightarrow \Sigma], \sqsubseteq)$

Ordnung auf Zustandstransformationen

Sogar...

Lemma 2

Das Paar $([\Sigma \rightarrow \Sigma], \sqsubseteq)$ ist eine vollständige partielle Ordnung (CPO) mit kleinstem Element \perp .

Weiter gilt: Die kleinste obere Schranke $\sqcup Y$ einer Kette Y ist gegeben durch

$$\text{graph}(\sqcup Y) = \cup \{ \text{graph}(g) \mid g \in Y \}$$

Das heißt: $(\sqcup Y) \sigma = \sigma' \iff \exists g \in Y. g \sigma = \sigma'$

Einschub: Graph einer Funktion

Der *Graph* einer totalen Funktion $f : M \rightarrow N$ ist definiert durch

$$\text{graph}(f) =_{df} \{ \langle m, n \rangle \in M \times N \mid f m = n \}$$

Es gilt:

- $\langle m, n \rangle \in \text{graph}(f) \wedge \langle m, n' \rangle \in \text{graph}(f) \Rightarrow n = n'$ (*rechtseindeutig*)
- $\forall m \in M. \exists n \in N. \langle m, n \rangle \in \text{graph}(f)$ (*linkstotal*)

Der *Graph* einer partiellen Funktion $f : M \rightarrow N$ mit Definitionsbereich $M_f \subseteq M$ ist definiert durch

$$\text{graph}(f) =_{df} \{ \langle m, n \rangle \in M \times N \mid f m = n \wedge m \in M_f \}$$

Vereinbarung...

Für $f : M \rightarrow N$ partiell definierte Funktion auf $M_f \subseteq M$ schreiben wir

- $f m = n$, falls $\langle m, n \rangle \in \text{graph}(f)$
- $f m = \text{undef}$, falls $m \notin M_f$

Stetigkeitsresultate (1)

Lemma 3

Sei $g_0 \in [\Sigma \rightarrow \Sigma]$, sei $p \in [\Sigma \rightarrow \mathbf{B}]$ und sei F definiert durch

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

Dann gilt: F ist stetig.

Zur Erinnerung: Seien (C, \sqsubseteq_C) und (D, \sqsubseteq_D) zwei CPOs und sei $f : C \rightarrow D$ eine Funktion von C nach D .

Dann heißt f ...

- *monoton* gdw. $\forall c, c' \in C. c \sqsubseteq_C c' \Rightarrow f(c) \sqsubseteq_D f(c')$
(Erhalt der Ordnung der Elemente)
- *stetig* gdw. $\forall C' \subseteq C. f(\bigsqcup_{C'} C') =_D \bigsqcup_{D} f(C')$
(Erhalt der kleinsten oberen Schranken)

Stetigkeitsresultate (2)

Lemma 4

Sei $g_0 \in [\Sigma \rightarrow \Sigma]$ und sei F definiert durch

$$F g = g \circ g_0$$

Dann gilt: F ist stetig.

Zusammen mit...

Lemma 5

Die Gleichungen zur Festlegung der denotationellen Semantik von WHILE (vgl. Folie 13 von heute) definieren eine totale Funktion

$$\llbracket \cdot \rrbracket_{ds} \in [\mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)]$$

...sind wir durch! Wir können beweisen:

$$\llbracket \cdot \rrbracket_{ds} : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

ist wohldefiniert!

Und somit wie anfangs angedeutet...

Aus...

1. Die Menge $[\Sigma \rightarrow \Sigma]$ der partiell definierten Zustandstransformationen bildet zusammen mit der Ordnung \sqsubseteq eine CPO.
2. Funktional F mit " $F g = \text{cond}(p, g, g_0)$ " und " $g \circ g_0$ " ist stetig
3. In der Definition von $\llbracket \cdot \rrbracket_{ds}$ wird die Fixpunktbildung ausschließlich auf stetige Funktionen angewendet.

...ergibt sich wie gewünscht:

$$\llbracket \cdot \rrbracket_{ds} : \mathbf{Prg} \rightarrow (\Sigma \rightarrow \Sigma_\varepsilon)$$

...ist wohldefiniert!

Vorschau auf die nächsten Vorlesungstermine...

- Di, 28.11.2006, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 05.12.2006, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 12.12.2006, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- *Di, 19.12.2006: Keine Vorlesung! (Ferialzeit)*
- *Di, 26.12.2006: Keine Vorlesung! (Ferialzeit)*

Vorschau auf die weiteren Vorlesungstermine...

- *Di, 02.01.2007: Keine Vorlesung! (Ferialzeit)*
- Di, 09.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 16.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 23.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 30.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1