Semantik

Mitschrift von www.kuertz.name

Hinweis: Dies ist **kein offizielles Script**, sondern nur eine private Mitschrift. Die Mitschriften sind teweilse **unvollständig, falsch oder inaktuell**, da sie aus dem Zeitraum 2001–2005 stammen. Falls jemand einen Fehler entdeckt, so freue ich mich dennoch über einen kurzen Hinweis per E-Mail – vielen Dank!

Klaas Ole Kürtz (klaasole@kuertz.net)

Inhaltsverzeichnis

1 Semantische Beschreibung von Programmen

1.3 Methoden der formalen Semantikdefinition

- *Operationelle Semantik:* genaue, schrittweise Abarbeitung auf einer hypothetischen Maschine; Sprache der Implementierer bzw. Übersetzerbauer
- *Denotationelle Semantik:* induktive, kompositionale Festlegung des E/A-Verhaltens; Sprache der Sprachdesigner und Theoretiker
- *Axiomatische Semantik:* Festlegung, wie Eigenschaften über Programme bewiesen werden können (Prädikattransformersemantik); Sprache der Algorithmiker und Programmierer

2 Gundlagen der denotationellen Semantik

2.1 Vollständige partielle Ordnungen, Monotonie, Stetigkeit

Sei eine geordnete Menge $(M, \leq)$ und $x \in M$ gegeben.

$$\begin{aligned} \textit{minimales Element } x &\iff \nexists\, y \in M : y \leq x \\ \textit{kleinstes Element } x &\iff \forall\, y \in M : x \leq y \end{aligned}$$

2.1.1 Definition (CPO)

Eine *CPO* ist eine geordnete Menge $(M, \leq)$ mit folgenden Eigenschaften

- Es gibt ein kleinstes Element $\bot \in M$.
- Jede Kette $\langle x^i \rangle_{i \in I} \subseteq M$ hat ein Supremum $\bigsqcup_{i \in I} x^{(i)}$ in M.

2.1.2 Definition (Flache Ordnung)

Eine *flache Ordnung* ist eine Ordnung, falls es $\bot \in M$ gibt mit

$$x \leq y \iff x = \bot \vee x = y$$

2.1.3 Definition (Monoton, stetig, strikt)

Seien $(M, \leq)$ und $(N, \leq)$ zwei CPOs. Eine Funktion $f : M \to N$ ist

- *strikt*, falls gilt:
$$f(\bot) = \bot$$
- *monoton*, falls für alle $x, y \in M$ gilt:
$$x \leq y \Longrightarrow f(x) \leq f(y)$$
- *stetig*, falls f monoton ist und für alle Ketten $\left\langle x^{(i)} \right\rangle_{i \in I}$ in M gilt:
$$f\left(\bigsqcup_{i \in I} x^{(i)}\right) = \bigsqcup_{i \in I} f(x^{(i)})$$

Die konstantwertige Funktion ist für jedes c einfach $\bar{c}(x) = c$.

2.1.4 Satz (Fundamentale Eigenschaften)

Seien CPOs $(M, \leq)$, $(N, \leq)$ und $(P, \leq)$ gegeben sowie Funktionen $f\colon M \to N$ und $g\colon N \to P$. Dann gilt:

- $(N, \leq)$ flach und f monoton, so f strikt oder konstant.
- $(M, \leq)$ flach und f strikt, so f monoton.
- In $(M, \leq)$ jede Kette stationär und f monoton, so f stetig.
- f und g monoton, stetig, strikt, so auch $g \circ f$ monoton, stetig, strikt.
- f monoton und Kette $\left\langle x^{(i)} \right\rangle_{i \in I}$ in M, so ist

$$\bigsqcup_{i \in I} f(x^{(i)}) \leq f\left(\bigsqcup_{i \in I} x^{(i)}\right)$$

2.1.6 Definition (Sub-CPO)

Sei $(M, \leq)$ eine CPO und $\emptyset \neq N \subseteq M$. Dann ist $(N, \leq|_N)$ eine *Sub-CPO*, falls gilt:

- $(N, \leq|_N)$ ist eine CPO.
- Für alle Ketten $\left\langle x^{(i)} \right\rangle_{i \in I}$ int N gilt: $\bigsqcup_{i \in I} x^{(i)} = \bigcup_{i \in I} x^{(i)}$. Dabei ist $\bigsqcup$ das Supremum bezüglich $\leq$; und $\bigcup$ das Supremum bezüglich $\leq|_N$.

2.1.7 Satz (Sub-CPO-Kriterium)

Sei $(M, \leq)$ eine CPO und $\emptyset \neq N \subseteq M$. Dann ist $(N, \leq|_N)$ eine *Sub-CPO* genau dann, wenn gilt:

- Es gibt in $(N, \leq|_N)$ ein kleinstes Element.
- Für alle Ketten $\left\langle x^{(i)} \right\rangle_{i \in I}$ int N gilt: $\bigsqcup_{i \in I} x^{(i)} \in N$, d.h. das Supremum liegt in N.

2.2 Semantische Algebren

2.2.1 Definition (Direktes Produkt von CPOs)

Seien CPOs $(M_i, \leq_i)$ für alle $1 \leq i \leq n$. Definiere

$$\langle x_1, \ldots, x_n \rangle \leq \langle y_1, \ldots, y_n \rangle :\Longleftrightarrow \forall\, 1 \leq i \leq n\colon x_i \leq_i y_i$$

Dann ist das *direkte Produkt* $(\prod_{i=1}^{n} M_i, \leq)$ eine CPO. Folgende *Projektionsfunktion* ist stetig:

$$pr_j \colon \prod_{i=1}^{n} M_i \to M_j \text{ mit } pr_j(\langle x_1, \ldots, x_n \rangle) = x_j$$

Merkregel:

$$\bigsqcup_{m \in I} \left\langle x_1^{(m)}, \ldots, x_n^{(m)} \right\rangle = \left\langle \bigsqcup_{m \in I} x_1^{(m)}, \ldots, \bigsqcup_{m \in I} x_n^{(m)} \right\rangle$$

2.2.3 Definition (Smash-Produkt von CPOs)

Seien CPOs $(M_i, \leq_i)$ für alle $1 \leq i \leq n$. Definiere

$$(M_{\otimes} :=) \quad \bigotimes_{i=1}^{n} M_i := \left(\prod_{i=1}^{n} (M_i \setminus \{\bot_i\}) \right) \cup \{\bot\}$$

Dann ist das *Smash-Produkt* $(M_{\otimes}, \leq \, |_{\otimes M})$ eine Sub-CPO des direkten Produkts. Die auf $M_{\otimes}$ eingeschränkte Projektionsfunktion ist stetig, ebenso die *Konstruktorfunktion cons*: $\prod_{i=1}^{n} M_i \to \bigotimes_{i=1}^{n} M_i$ mit

$$cons(\langle x_1, \ldots, x_n \rangle) = \begin{cases} \langle x_1, \ldots, x_n \rangle & \text{falls } \forall\, i \colon x_i \neq \bot_i \\ \langle \bot_1, \ldots, \bot_n \rangle & \text{falls } \exists\, i \colon x_i = \bot_i \end{cases}$$

2.2.6 Definition (Separierte Summe von CPOs)

Seien CPOs $(M_i, \leq_i)$ gegeben. Die *separierte Summe* ist die Menge

$$\sum_{i=1}^{n} M_i = \left(\bigcup_{i=1}^{n} (M_i \times \{i\}) \right) \cup \{\bot\}.$$

Die *Summenordnung* ist definiert als:

$$x \leq y :\Longleftrightarrow (x = \bot) \ \vee \ (x = \langle \hat{x}, i \rangle \wedge y = \langle \hat{y}, i \rangle \wedge \hat{x} \leq_i \hat{y})$$

Dann ist die separierte Summe mit Summenordnung wieder eine CPO. Die folgenden *Injektionsfunktionen* ist stetig:

$$in_j \colon M_j \to \sum_{i=1}^{n} M_i \text{ mit } in_j(x) = \langle x, j \rangle$$

Auch die *Projektionsfunktionen* sind stetig:

$$pr_j \colon \sum_{i=1}^{n} M_i \to M_j \text{ mit } pr_j(x) = \begin{cases} u & \text{falls} \quad x = \langle u, j \rangle \\ \bot_j & \text{sonst} \end{cases}$$

2.2.8 Definition (Verschmolzene Summe)

Seien CPOs $(M_i, \leq_i)$ gegeben. Die *verschmolzene Summe* ist die Menge

$$\bigoplus_{i=1}^{n} M_i = \left(\bigcup_{i=1}^{n} ((M_i \setminus \{\bot_i\}) \times \{i\}) \right) \cup \{\bot\}.$$

Dann ist die verschmolzene Summe mit Summenordnung wieder eine CPO. Die Projektionsfunktionen sind weiter stetig, die *Injektionsfunktionen* ebenfalls:

$$in_j \colon M_j \to \sum_{i=1}^{n} M_i \text{ mit } in_j(x) = \begin{cases} \bot & \text{falls} \quad x = \bot_j \\ \langle x, j \rangle & \text{sonst} \end{cases}$$

2.2.9 Definition (Funtkons-CPO)

Seien $(M, \leq)$ und $(N, \leq)$ zwei CPOs. Definiere die *Funktionsordnung* auf der Menge N^M der Funktionen von N nach M:

$$f \leq g :\Longleftrightarrow \forall\, x \in M \colon f(x) \leq g(x)$$

Dann ist $(N^M, \leq)$ wieder eine CPO mit kleinstem Element Ω:

$$\Omega \colon M \to N \text{ mit } \Omega(x) = \bot$$

2.2.11 Satz (Abschlußeigenschaften bei Funktionen)

Das Supremum einer Kette von monotonen, stetigen, strikten Funktionen ist wiederum monoton, stetig, strikt.

2.2.12 Definition und Satz (Sub-CPOs von Funktionen)

Seien CPOs $(M, \leq)$ und $(N, \leq)$ gegeben. Dann sind folgende Mengen Sub-CPOs:

$$\begin{aligned} (M \to N) &:= \left\{ f \in N^M \;\middle|\; f \text{ monoton} \right\} \\ [M \to N] &:= \left\{ f \in N^M \;\middle|\; f \text{ stetig} \right\} \\ (M \to_s N) &:= \left\{ f \in N^M \;\middle|\; f \text{ monoton, strikt} \right\} \\ [M \to_s N] &:= \left\{ f \in N^M \;\middle|\; f \text{ stetig, strikt} \right\} \end{aligned}$$

Es gilt folgende Beziehung:

$$\begin{array}{ccc} [M \to_s N] & \longrightarrow & [M \to N] \\ \downarrow & & \downarrow \\ (M \to_s N) & \longrightarrow & (M \to N) \\ & \searrow \quad \swarrow & \\ & N^M & \end{array}$$

Curryfizierung und Umkehrfunktion:

$$\begin{array}{rl} curry\colon & P^{M\times N} \to (P^N)^M \text{ mit } curry(f)(x)(y) = f(x,y) \\ uncurry\colon & (P^N)^M \to P^{M\times N} \text{ mit } uncurry(f)(x,y) = f(x)(y) \end{array}$$

2.2.13 Definition und Satz (Lifting)

Übergang von M zu $M^\perp := M \cup \{\perp\}$:

- Lifting erster Art: $(M, =) \to (M^\perp, \leq)$ mit flacher Ordnung $\leq$
- Lifting zweiter Art: $(M, \leq) \to (M^\perp, \leq^\perp)$ mit modifizierter Ordnung $\leq^\perp$, so daß $\perp \leq^\perp x$ ist für alle $x \in M^\perp$

2.3 Fixpunkttheorie

2.3.1 Beispiel (Rechenvorschriften und Fixpunktgleichungen)

Gesucht: Fixpunkt $f \in M^M$ des Funktionals $\tau\colon M^M \to M^M$, d.h. $\tau[f] = f$.

2.3.2 Definition (kleinster Fixpunkt)

Falls in der Menge der Fixpunkte einer Funktion ein kleinstes Element existiert, so heißt dies μ_f:

$$x = \mu_f :\Longleftrightarrow (f(y) = y \to x \leq y)$$

2.3.3 Satz (Funktionsiteration von unten)

Sei $(M, \leq)$ eine CPO, $f \in (M \to M)$. Existiert μ_f, so ist

$$\bigsqcup_{i\geq 0} f^{(i)}(\perp) \leq \mu_f \quad \Longleftrightarrow \quad \perp \leq f(\perp) \leq f(f(\perp)) \leq f(f(f(\perp))) \leq \ldots \leq \mu_f$$

2.3.4 Satz (Fixpunktsatz von Knaster, Tarski und Kleene

) Ist $(M, \leq)$ CPO und $f \in [M \to M]$, so existiert μ_f und es gilt:

$$\mu_f = \bigsqcup_{i\geq 0} f^{(i)}(\perp)$$

2.3.6 Definition (stetige Prädikate)

Sei $(M, \leq)$ CPO und $\left\langle x^{(i)} \right\rangle_{i \in I}$ Kette. $P : M \to \mathbb{B}$ ist ein *stetiges Prädikat*, falls gilt:

$$\forall\, x\colon \left(P(x^{(i)}) = \mathtt{tt} \implies P\left(\bigsqcup_{i \in I} x^{(i)} \right) = \mathtt{tt} \right)$$

2.3.7 Satz (Berechnungsinduktion)

Sei $(M, \leq)$ CPO, $f \in [M \to M]$, $P\colon M \to \mathbb{B}$ stetig. Dann gilt $P(\mu_f) = \mathtt{tt}$ unter folgender Voraussetzung:

$$P(\bot) = \mathtt{tt} \ \wedge\ (\forall\, x \in M\colon\ (P(x) = \mathtt{tt} \implies P(f(x)) = \mathtt{tt}))$$

2.3.8 Satz (Syntaktisches Kriterium für stetige Prädikate)

Sei $(M, \leq)$ CPO. Ein Prädikat $P\colon M \to \mathbb{B}$ ist stetig, falls eine CPO $(N, \leq)$ und Funktionen $f, g \in M \to N$ existieren mit

$$\forall\, x\colon\ (P(x) = \mathtt{tt} \iff f(x) \leq g(x))$$

2.3.9 Satz (Konjunktion stetiger Prädikate)

Seien $P, Q\colon M \to \mathbb{B}$ stetige Prädikate. Dann ist auch die Konjunktion $(P \wedge Q)$ stetig.

2.3.11 Satz (Lemma von Park)

Sei $(M, \leq)$ CPO, $x_0 \in M$ und $f \in [M \to M]$ mit $f(x_0) \leq x_0$, so gilt $\mu_f \leq x_0$.

2.3.12 Lemma von Zorn

Sei $(M, \leq)$ geordnete Menge. Hat jede Kette in M eine obere Schranke, so besitzt M ein maximales Element.

2.3.13 Satz (Fixpunktsatz und Berechnungsinduktion, monotoner Fall)

Jede monotone Funktion auf einer CPO besitzt einen kleinsten Fixpunkt. Das Prinzip der Berechnungsinduktion gilt auch für monotone Funktionen.

2.3.14 Beispiel (Funktionsiteration bei monotonen Funktionen)

Die *Egli-Milner-Ordnung*: Sei M eine flach geordnete Menge. Sei $\mathbf{P}(M) := 2^M \setminus \{\emptyset\}$. Seien $X, Y \in \mathbf{P}(M)$.

$$\begin{aligned} X \leq_{\mathrm{EM}} Y \quad &:\Longleftrightarrow \quad (\forall\, x \in X\ \exists\, y \in Y \colon x \leq y) \wedge (\forall\, y \in Y\ \exists\, x \in X \colon x \leq y) \\ &\Longleftrightarrow \quad \begin{cases} X \setminus \{\bot\} \subseteq Y & \text{falls} \quad \bot \in X \\ X = Y & \text{falls} \quad \bot \notin X \end{cases} \end{aligned}$$

Dann ist $(\mathbf{P}(M), \leq_{\mathrm{EM}})$ die Plotkin-Potenzmengenbereich-CPO mit kleinstes Element $\{\bot\}$ und Supremum wie folgt:

$$\bigsqcup_{i \in I} X^{(i)} = \begin{cases} \bigcup_{i \in I} X^{(i)} & \text{falls } \forall\, i \in I \colon \bot \in X^{(i)} \\ X^{(j)} & \text{falls } \bot \notin X^{(j)} \end{cases}$$

2.3.15 Satz (Eigenschaften des Fixpunktoperators)

Der Fixpunktoperater $\mu \colon M^M \to M$, definiert durch $\mu(f) = \mu_f$, ist monoton auf den monotonen und stetig auf den stetigen Funktionen:

μ	M^M	$(M \to M)$	$[M \to M]$
total	✗	✓	✓
monoton	✗	✓	✓
stetig	✗	✗	✓

3 Denotationelle Semantik applikativer Sprachen

3.1 Die Beispielsprache

3.1.1 Syntaktische Basis von $\mathcal{L}_A$

(1) *Signatur* $\Sigma = (S, K, F)$ mit

  (a) *Sorten* S (mindestens $\texttt{bool} \in S$)

  (b) *Konstanten* K (aus Sorten, d.h. $K = \biguplus K_s$ für $s \in S$)

  (c) *Funktionen* F (getypt: $F_{(a,r)}$ mit $a \in S^+$ und $r \in S$)

(2) *Typen* T (induktiv definiert über die Sorten):

  (a) Sorten $S \subseteq T$

  (b) *Produkttypen* $(m_1; \ldots, m_k) \in T$ für alle $m_i \in T$

  (c) *Funktionstypen* $(m \to n) \in T$ für $m, n \in T$

(3) *frei wählbare Bezeichner* X (getypt, d.h. $X = \biguplus X_m$ mit $m \in T$)

3.1.2 Terme (Programme von $\mathcal{L}_A$)

Familie $\langle \mathsf{EXP}_m \rangle_{m \in T}$ der *Terme* bzw. $\mathcal{L}_A$*-Programme*:

(1) *frei wählbare Bezeichner*: $X_m \subseteq \mathsf{EXP}_m$

(2) *Konstanten*: $K_m \in \mathsf{EXP}_m$

(3) *Basisoperationen*: $F_{a_1 \cdots a_n, r} \subseteq \mathsf{EXP}_{a_1;\ldots;a_n \to r}$

(4) *Tupel*: $(t_1; \ldots; t_n) \in \mathsf{EXP}_{m_1;\ldots;m_n}$

(5) *Alternative*: $\texttt{if}\ b\ \texttt{then}\ t_1\ \texttt{else}\ t_2\ \texttt{fi} \in \mathsf{EXP}_m$

(6) *Funktionsapplikation*: $(e(t_1; \ldots; t_n)) \in \mathsf{EXP}_m$

(7) *Funktionsabstraktion*: $((x_1 \colon m_1; \ldots; x_n \colon m_n) \colon m; t) \in \mathsf{EXP}_{m_1;\ldots;m_n \to m}$

(8) *Rekursion*: $\texttt{fun}\ x = t \in \mathsf{EXP}_m$

3.2 Denotationelle Semantik der Beispielsprache

3.2.1 Interpretation der Basis-Signatur

Basis-Signatur $\Sigma = (S, K, F)$ wird interpretiert durch eine Σ-Algebra

$$A = (\langle s^A \rangle_{s \in S}, \langle c^A \rangle_{c \in K}, \langle f^A \rangle_{f \in F})$$

(1) s^A ist nichtlere *Trägermenge*, $\texttt{bool}^A = \mathbb{B}$

(2) c^A ist ein Element aus $\bigcup_{s \in S} S^A$

(3) f^A ist partielle Funktion über den Trägermengen von A

3.2.2 Interpretation der Typen

Jedem Typ $m \in T$ wird eine CPO $(i[m], \leq)$ zugeordnet:

(1) Sorte m:

$$i[m] = m^A \cup \{\bot_m\} \quad \text{mit flacher Ordnung}$$

(2) Produkttyp $(m_1; \ldots; m_n)$:

$$i[m_1; \ldots; m_n] = \prod_{j=1}^{k} i[m_j] \quad \text{mit Produktordnung}$$

(3) Funktionstyp $m_1 \to m_2$:

$$i[m_1 \to m_2] = (i[m_1] \to i[m_2]) \quad \text{mit Funktionsordnung}$$

3.2.3 Strikte Erweiterung der Basisoperationen

Jeder Funktion $f \in F_{a_1 \cdots a_n; r}$ wird ihre *strikte Erweiterung* zugeordnet:

$$f^\bot(u_1, \ldots, u_n) = \begin{cases} \bot & \text{falls} \quad (\exists\, j \colon u_j = \bot) \\ & \vee \quad (f^A(u_1, \ldots, u_k) \text{ undef.}) \\ f^A(u_1, \ldots, u_k) & \text{sonst} \end{cases}$$

3.2.4 Satz

Die strikte Erweiterung $f^\bot$ einer Funktion f ist stetig.

3.2.5 Definition (Umgebung)

Definiere die Menge ENV *(Umgebung)* als Menge der folgenden *Zuordnungen*:

$$a\colon X \to \bigcup_{m\in T}^{i[m]}$$

Abänderung einer Zuordnung $a \in \mathsf{ENV}$ an einer Stelle $x \in X_m$ ist

$$a[x \leftarrow u] \in \mathsf{ENV}$$

3.2.6 Semantik von $\mathcal{L}_A$

Semantikfunktion

$$[\![\cdot]\!]\colon \mathsf{EXP}_m \to i[m]^{ENV} \quad \text{so daß} \quad [\![t]\!]\colon \mathsf{ENV} \to i[m]$$

(1) frei wählbare Bezeichner: $[\![x]\!](a) = a(x)$

(2) Konstanten: $[\![c]\!](a) = c^A$

(3) Basisoperationen: $[\![f]\!](a) = f^\perp$

(4) Tupel: $[\![(t_1;\ldots;t_k)]\!](a) = \langle [\![t_1]\!](a), \ldots, [\![t_k]\!](a)\rangle$

(5) Alternative:

$$[\![\texttt{if}\ b\ \texttt{then}\ t_1\ \texttt{else}\ t_2\ \texttt{fi}]\!](a) = \begin{cases} [\![t_1]\!](a) \text{ falls } [\![b]\!](a) = \texttt{tt} \\ [\![t_2]\!](a) \text{ falls } [\![b]\!](a) = \texttt{ff} \\ \perp \quad \text{falls } [\![b]\!](a) = \perp \end{cases}$$

(6) Funktionsapplikation:

$$[\![t(t_1,\ldots,t_k)]\!](a) = [\![t]\!](a)\,([\![t_1]\!](a), \ldots, [\![t_k]\!](a))$$

(7) Funktionsabstraktion:

$$[\![(x_1\colon m_1;\ldots;x_k\colon m_k)\colon n; t]\!](a) = h$$

Hierbei ist

$$h \in i[m_1;\ldots;m_k \to n] \text{ mit } h(u_1,\ldots,u_k) = [\![t]\!](a[x_i \leftarrow u_i])$$

Damit ist

$$[\![(x_1\colon m_1;\ldots;x_k\colon m_k)\colon n; t]\!](a)(u_1,\ldots,u_k) = [\![t]\!](a[x_i \leftarrow u_i])$$

(8) Rekursion:

$$[\![\texttt{fun}\ x = t]\!](a) = \mu_\tau$$

Hierbei ist

$$\tau \in i[m] \to i[m] \text{ mit } \tau[u] = [\![t]\!](a[x \leftarrow u])$$

3.2.7 Satz (Monotonie der Semantik)

Sei t Term von $\mathcal{L}_A$, $y \in X_m$, $a \in \mathsf{ENV}$, $u, v \in i[m]$. Dann ist:

$$u \leq v \Longrightarrow [\![t]\!](a[y \leftarrow u]) \leq [\![t]\!](a[y \leftarrow v])$$

3.2.8 Satz (Stetigkeit der Semantik)

Sei t Term von $\mathcal{L}_A$, $y \in X_m$, $a \in \mathsf{ENV}$, $\left\langle u^{(i)} \right\rangle_{i \in I} \in i[m]$. Dann gilt:

$$[\![t]\!] \left(a \left[y \leftarrow \bigsqcup_{i \in I} u^{(i)} \right] \right) = \bigsqcup_{i \in I} [\![t]\!](a[y \leftarrow u^{(i)}])$$

3.2.9 Satz (Koinzidenzlemma)

Stimmen zwei Zuordnungen a_1 und a_2 auf allen in einem $\mathcal{L}_A$-Term t frei vorkommenden Bezeichnern überein, so ist $[\![t]\!](a_1) = [\![t]\!](a_2)$.

3.2.10 Beispiel (Semantik einer Rechenvorschrift)

Gegen sei:

$$\begin{array}{l} \texttt{fun}\ F = (x\colon nat)\colon nat; \\ \quad \texttt{if}\ x = 0\ \texttt{then}\ 1 \\ \qquad\qquad \texttt{else}\ y * F(x-1)\ \texttt{fi} \end{array}$$

Die Semantik:

$$[\![\texttt{fun}\ F = \ldots]\!](a)(u) = \begin{cases} a(y)^u & \text{falls} \quad u \neq \bot \wedge a(y) \neq \bot \\ \bot & \text{falls} \quad u = \bot \vee a(y) = \bot \end{cases}$$

3.3 Parameterübergabe in der Beispielsprache

Unterscheidung *call-by-value* gegenüber *call-by-name*

3.3.1 Beispiel (Parameterübergabemechanismus)

Betrachte folgende Funktion:

$$\texttt{fun}\ F = (x\colon int)\colon int; 1(1 \div 0)$$

Dann ergibt sich hier die *call-by-name*-Semantik

$$[\![\texttt{fun}\ F = (x\colon int)\colon int; 1(1 \div 0)]\!](a) = \bar{1}(\bot) = 1$$

3.4 Eine alternative Semantikdefinition

3.4.1 Alternative Interpretation der Typen

(2') Produkttyp $(m_1; \ldots; m_n)$:

$$i[m_1; \ldots; m_n] = \bigotimes_{j=1}^{k} i[m_j] \quad \text{mit Produktordnung}$$

(3') Funktionstyp $m_1 \to m_2$:

$$i[m_1 \to m_2] = (i[m_1] \to i[m_2])^{\perp} \quad \text{mit Funktionsordnung}$$

Funktionsordnung:

$$f \leq^{\star} g \iff f = \perp \vee (f \neq \perp \wedge g \neq \perp \wedge f \leq g)$$

3.4.2 Alternative Festlegung der Semantikfunktion

(4') Tupel:

$$[\![(t_1; \ldots; t_k)]\!](a) = cons(\langle [\![t_1]\!](a), \ldots, [\![t_k]\!](a)\rangle)$$

Hierbei ist

$$cons(\langle x_1, \ldots, x_n\rangle) = \begin{cases} \langle x_1, \ldots, x_n\rangle & \text{falls} \quad \forall\, i\colon x_i \neq \perp_i \\ \langle \perp_1, \ldots, \perp_n\rangle & \text{falls} \quad \exists\, i\colon x_i = \perp_i \end{cases}$$

(6') Funktionsapplikation:

$$[\![t(t_1, \ldots, t_k)]\!](a) = \begin{cases} \perp & \text{falls} \quad [\![t]\!](a) = \perp \\ [\![t]\!](a)\,(cons([\![t_1]\!](a), \ldots, [\![t_k]\!](a))) & \text{falls} \quad [\![t]\!](a) \neq \perp \end{cases}$$

(7') Funktionsabstraktion:

$$[\![(x_1\colon m_1; \ldots; x_k\colon m_k)\colon n; t]\!](a) = h[\langle \perp_1, \ldots, \perp_k\rangle \leftarrow \perp]$$

Hierbei ist

$$h \in i[m_1; \ldots; m_k \to n] \text{ mit } h(u_1, \ldots, u_k) = [\![t]\!](a[x_i \leftarrow u_i])$$

Damit hat sich der Funktionsbereich von h verändert von $\prod$ zu $\bigotimes$.

3.4.3 Beispiel (Unterschiede zwischen den Semantiken)

Verwende wieder folgende Funktion:

$$\texttt{fun}\ F = (x\colon int)\colon int; 1(1 \div 0)$$

Dann ergibt sich mit der abgeänderten Semantik die *call-by-value*-Semantik

$$[\![\texttt{fun}\ F = (x\colon int)\colon int; 1(1 \div 0)]\!](a) = \bot$$

3.5 Fragen, die sich beim Sprachausbau ergeben

3.5.1 Systeme von verschränkt rekursiven Recehenvorschriften

Erweiterung der Syntax:

(8') verschränkte Rekursion:

$$\texttt{fun}\ x_1, \ldots, x_k = t_1, \ldots, t_k \in \mathsf{EXP}_{m_1;\ldots;m_k}$$

Änderung der Semantik:

(8') Es gilt

$$[\![\texttt{fun}\ x_1, \ldots, x_k = t_1, \ldots, t_k]\!](a) = \mu_\tau$$

Hierbei ist τ folgende Funktion:

$$\tau\colon \prod_{j=1}^{k} i[m_j] \to \prod_{j=1}^{k} i[m_j]$$

wobei

$$\tau[h_1, \ldots, h_k] = \langle [\![t_1]\!](a[x_i \leftarrow h_i]), \ldots, [\![t_k]\!](a[x_i \leftarrow h_i])\rangle$$

3.5.2 Deklarationen und Abschnitte

3.5.3 Spezifikationselemente

3.5.4 Nichtdeterminismus

4 Denotatonelle Semantik prozeduraler Sprachen

4.1 Die Beispielsprache

4.1.1 Syntaktische Basis von $\mathcal{L}_P$

(1) *Signatur* $\Sigma = (S, K, F)$ genau wie bei $\mathcal{L}_A$ – d.h. mindestens mit einer Sorte `bool`.

(2) *frei wählbare Bezeichner*[1] X (getypt, d.h. $X = \biguplus X_m$ mit $m \in S$) – x heißt hier *Programmvariable* (als Unterscheidung zu Variablen in der Logik, beachte referentielle Transparenz)

4.1.2 Terme

Familie $\langle \mathsf{SEXP}_m \rangle_{m \in S}$ der *Terme* jeweils einer Sorte $m \in S$:

(1) *Programmvariable*: $X_m \subseteq \mathsf{SEXP}_m$

(2) *Konstanten*: $K_m \in \mathsf{SEXP}_m$

(3) *Applikation* einer Basisoperationen: falls $f \in F_{(m_1 \ldots m_k, m)}$ und $t_i \in \mathsf{SEXP}_{m_i}$, dann $f(t_1, \ldots, t_k) \in \mathsf{SEXP}_m$

4.1.3 Anweisungen (Programme von $\mathcal{L}_P$)

Menge STAT der Anweisungen über der Basis von $\mathcal{L}_P$ *($\mathcal{L}_P$-Programme):*

(1) *Leere Anweisung*:

$$\texttt{skip} \in \mathsf{STAT}$$

(2) *Undefinierte Anweisung*:

$$\texttt{abort} \in \mathsf{STAT}$$

(3) *Zuweisung*:

$$(x := t) \in \mathsf{STAT}$$

(4) *Bedingte Anweisung* oder *Alternative*:

$$\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{fi} \in \mathsf{STAT}$$

[1] zu funktionalen Sprachen: „. . . der Wert einer Variable ist unabhängig vom Ort, von der Zeit, vom Wetter, . . .“

(5) *Schleife*:

$$\texttt{while}\ b\ \texttt{do}\ s\ \texttt{od} \in \mathsf{STAT}$$

(6) *Sequentielle Komposition*: für

$$s_1, s_2 \in \mathsf{STAT} \implies (s_1; s_2) \in \mathsf{STAT}$$

4.1.4 Bemerkung zur undefinierten Anweisung

`abort` ist Vertreter der Klasse aller Anweisungen, deren Ausführung einen Fehler erzeugt, z.B. $x := 1 \div 0$ – die Anweisung `abort` wurde nur aus methodischen Gründen eingeführt.

4.2 Denotationelle Semantik der Beispielsprache

4.2.1 Interpretation der Basis-Signatur

Die Basis-Signatur $\Sigma = (S, K, F)$ wird interpretiert durch eine Σ-Algebra analog zur Interpretation bei $\mathcal{L}_A$, siehe 3.2.1:

$$A = (\langle s^A \rangle_{s \in S}, \langle c^A \rangle_{c \in K}, \langle f^A \rangle_{f \in F})$$

Wichtig ist wiederum $\texttt{bool}^A = \mathbb{B} = \{\texttt{tt}, \texttt{ff}\}$.

4.2.2 Interpretation der Sorten

Jeder Sorte $m \in S$ wird die Menge $i[m] := (m^A)^\perp$ zugeordnet mit der flachen Ordnung.

4.2.3 Definition (Speicher)

STOREist Lifting erster Art der Menge aller foFunktionen

$$\sigma \colon X \to \bigcup_{m \in S} i[m]$$

Somit $\sigma(x) \in i[m]$. Das Element $\perp \in \mathsf{STORE}$ heißt *undefinierter Speicher*; alle Funktionen $\sigma \colon X \to \bigcup i[m]$ heißen *definierte Speicher*.

Abänderung eines definierten Speichers $\sigma \in \mathsf{STORE}$ an der Stelle $x \in X_m$ zu $u \in i[m]$:

$$\sigma[x \leftarrow u](y) = \begin{cases} \sigma(y) & \text{falls} \quad y \neq x \\ u & \text{falls} \quad y = x \end{cases}$$

4.2.4 Semantik von $\mathcal{L}_P$: Terme

Die Semantik ist eine Funktion

$$[\![\cdot]\!] \colon \mathsf{SEXP}_m \to i[m]^{\mathsf{STORE}},$$

die zu jedem Term $t \in \mathsf{SEXP}_m$ eine Funktion $[\![t]\!] \colon \mathsf{STORE} \to i[m]$ definiert:

(1) Programmvariablen: Für $x \in X_m$ ist

$$[\![x]\!](\sigma) = \begin{cases} \sigma(x) & \text{falls} \quad \sigma \neq \bot \\ \bot_m & \text{falls} \quad \sigma = \bot \end{cases}$$

(2) Konstanten:

$$[\![c]\!](\sigma) = \begin{cases} c^A & \text{falls} \quad \sigma \neq \bot \\ \bot_m & \text{falls} \quad \sigma = \bot \end{cases}$$

(3) Applikation einer Basisoperation:

$$[\![f(t_1, \ldots, t_k)]\!](\sigma) = \begin{cases} f^{\bot}([\![t_1(\sigma), \ldots, [\![t_k]\!](\sigma)]\!]) & \text{falls} \quad \sigma \neq \bot \\ \bot_m & \text{falls} \quad \sigma = \bot \end{cases}$$

4.2.5 Satz (Stetigkeit und Striktheit der Termsemantik)

Für alle Terme $t \in \mathsf{SEXP}_m$ gilt:

$$[\![t]\!] \in [\mathsf{STORE} \to_s i[m]]$$

4.2.6 Semantik von $\mathcal{L}_P$: Anweisungen

Die Semantikfunktion

$$[\![\cdot]\!] \colon \mathsf{STAT} \to \mathsf{STORE}^{\mathsf{STORE}}$$

ordnet jeder Anweisung $s \in \mathsf{STAT}$ eine Funktion $[\![s]\!] \colon \mathsf{STORE} \to \mathsf{STORE}$ zu:

(1) Leere Anweisung:

$$[\![\texttt{skip}]\!](\sigma) = \sigma$$

(2) Undefinierte Anweisung:

$$[\![\texttt{abort}]\!](\sigma) = \bot$$

(3) Zuweisung:

$$[\![x := t]\!](\sigma) = \begin{cases} \sigma[x \leftarrow [\![t]\!](\sigma)] & \text{falls} \quad [\![t]\!](\sigma) \neq \bot_m \\ \bot & \text{falls} \quad [\![t]\!](\sigma) = \bot_m \end{cases}$$

(4) Bedingte Anweisung oder Alternative:

$$[\![\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{fi}]\!](\sigma) = \begin{cases} [\![s_1]\!](\sigma) & \text{falls} \quad [\![b]\!](\sigma) = \texttt{tt} \\ [\![s_2]\!](\sigma) & \text{falls} \quad [\![b]\!](\sigma) = \texttt{ff} \\ \bot & \text{falls} \quad [\![b]\!](\sigma) = \bot_{\texttt{bool}} \end{cases}$$

(5) Schleife:

$$[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{od}]\!](\sigma) = \mu_\tau(\sigma)$$

Dabei ist das Funktional

$$\tau \colon \mathsf{STORE}^{\mathsf{STORE}} \to \mathsf{STORE}^{\mathsf{STORE}}$$

punktweise definiert für $h \colon \mathsf{STORE} \to \mathsf{STORE}$ und $\varrho \in \mathsf{STORE}$ durch

$$\tau[h](\varrho) = \begin{cases} h([\![s]\!](\varrho)) & \text{falls} \quad [\![b]\!](\varrho) = \texttt{tt} \\ \varrho & \text{falls} \quad [\![b]\!](\varrho) = \texttt{ff} \\ \bot & \text{falls} \quad [\![b]\!](\varrho) = \bot_{\texttt{bool}} \end{cases}$$

(6) Sequentielle Komposition: für

$$[\![s_1; s_2]\!](\sigma) = [\![s_2]\!]([\![s_1]\!](\sigma))$$

Die Semantik von `while` läßt sich „strecken“:

$$[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{od}]\!] = [\![\texttt{if}\ b\ \texttt{then}\ s; \texttt{while}\ b\ \texttt{do}\ s\ \texttt{od}\ \texttt{else}\ \texttt{skip}\ \texttt{fi}]\!]$$

4.2.7 Satz (Stetigkeit des Schleifenfunktionals)

Das für die Definition der Semantik der `while`-Schleife verwendete Funktional τ ist monoton stetig, d.h.

$$\tau \in \left[\mathsf{STORE}^{\mathsf{STORE}} \to \mathsf{STORE}^{\mathsf{STORE}}\right]$$

4.2.8 Satz (Stetigkeit und Striktheit der Anweisungssematik)

Für alle Anweisungen $s \in \mathsf{STAT}$ gilt die Beziehung

$$[\![s]\!] \in [\mathsf{STORE} \to_s \mathsf{STORE}]$$

4.2.9 Beispiel (Herunterzählen einer Schleife)

$$[\![\texttt{while}\ x \neq 0\ \texttt{do}\ x := x - 1\ \texttt{od}]\!](\sigma)$$
$$= \begin{cases} \sigma[x \leftarrow 0] & \text{falls} \quad \sigma \neq \bot \wedge \sigma(x) \neq \bot_{\texttt{nat}} \\ \sigma & \text{sonst} \end{cases}$$

4.3 Fragen, die sich beim Sprachausbau ergeben

Weitere Anweisungstypen (`for`, `switch`, ...) werden zurückgeführt auf die bisherigen Anweisungen. *Sprünge* werden hier nicht behandelt. Andere Erweiterungen in den nächsten Kapiteln...

4.3.1 Eingabe und Ausgabe

Erweiterung der Syntax:

(7) *Eingabe*: für alle $x \in X$ sei

$$\texttt{read}\,(x) \in \mathsf{STAT}$$

(8) *Ausgabe*: für alle $t \in \mathsf{SEXP}_m$ sei

$$\texttt{write}\,(x) \in \mathsf{STAT}$$

Zur Semantik: definiere CPOs IN und OUT als Lifting erster Art der Menge der endlichen Sequenzen mit Elementen aus den Trägermengen:

$$\mathsf{IN} = \mathsf{OUT} = \left((\bigcup_{m \in S} m^A)^* \right)^\perp$$

Definiere nun Zustände bestehend aus noch zu lesender Eingabe, Speicher und schon geschriebener Ausgabe:

$$\mathsf{STATE} = \mathsf{IN} \otimes \mathsf{STORE} \otimes \mathsf{OUT}$$

Für Terme $t \in \mathsf{SEXP}_m$ und Anweisungen $s \in \mathsf{STAT}$ ändert sich nun die Semantik in natürlicher Weise:

$$[\![t]\!] \colon \mathsf{STATE} \to i[m] \qquad [\![s]\!] \colon \mathsf{STATE} \to \mathsf{STATE}$$

Semantik für die neuen Anweisungen:

(7) *Eingabe*: $[\![\texttt{read}\,(x)]\!](\langle i, \sigma, o\rangle)$

$$= \begin{cases} \langle \mathrm{rest}(i), \sigma[x \leftarrow \mathrm{first}(i)], o\rangle & \text{falls} \quad \substack{\langle i,\sigma,o\rangle \neq \langle \perp,\perp,\perp\rangle \\ \wedge\, i \neq \varepsilon \,\wedge\, \mathrm{first}(i) \in m^A} \\ \langle \perp, \perp, \perp\rangle & \text{sonst} \end{cases}$$

(8) *Ausgabe*: $[\![\texttt{write}\,(t)]\!](\langle i, \sigma, o\rangle)$

$$= \begin{cases} \langle i, \sigma, \mathrm{append}(o, [\![t]\!](\langle i, \sigma, o\rangle))\rangle & \text{falls} \quad \substack{\langle i,\sigma,o\rangle \neq \langle \perp,\perp,\perp\rangle \\ \wedge\, [\![t]\!](\langle i,\sigma,o\rangle) \neq \perp_m} \\ \langle \perp, \perp, \perp\rangle & \text{sonst} \end{cases}$$

4.3.2 Deklarationen und Blockstrukturen

(9) *Blöcke*: $x \in X_m$, $t \in \mathsf{SEXP}_m$ und $s \in \mathsf{STAT}$

$$\texttt{begin var}\ x\colon m := t; s\ \texttt{end} \in \mathsf{STAT}$$

Vorgehensweise:

(i) ENV besteht aus Zuordnungen $a\colon X \to \mathsf{LOC}$

(ii) STORE ist Lifting erster Art der Menge von Abbildungen

$$\sigma\colon \mathsf{LOC} \to \left(\bigcup_{m\in S} i[m]\right) \cup \{\texttt{free}\}$$

(iii) Semantikfunktionen ändern sich:

$$[\![t]\!]\colon \mathsf{ENV} \to [\mathsf{STORE} \to_s i[m]] \qquad [\![s]\!]\colon \mathsf{ENV} \to [\mathsf{STORE} \to_s \mathsf{STORE}]$$

Umsetzung:

(i) Semantik eines Blocks:

(9) *Blöcke*:

$$[\![\texttt{begin var}\ x\colon m := t; s\ \texttt{end}]\!](a)(\sigma)$$
$$= \begin{cases} [\![s]\!](a[x \leftarrow l_\sigma])(\sigma[l_\sigma \leftarrow [\![t]\!](a)(\sigma)]) & \text{falls} \quad [\![t]\!](a)(\sigma) \neq \bot_m \\ \bot & \text{sonst} \end{cases}$$

(ii) Umsetzung mittels

$$[\![s_1; s_2]\!](a)(\sigma) = [\![s_2]\!](a)([\![s_1]\!](a)(\sigma))$$

(iii) klar, nur als Beispiel:

$$[\![x]\!](a)(\sigma) = \begin{cases} \sigma(a(x)) & \text{falls} \quad \sigma \neq \bot \\ \bot & \text{falls} \quad \sigma = \bot \end{cases}$$

Beispiel:

```
begin varx: bool := true;
    begin varx: bool := not(x);
        y := x
    end;
    z := x
end;
```

Erweiterung auf mehrere Variablen:

$$\begin{aligned} & [\![\texttt{begin var}\ x_1\colon m_1 := t_1; \ldots; x_m\colon m_m := t_m; s\ \texttt{end}]\!] \\ = \ & [\![\texttt{begin var}\ x_1\colon m_1 := t_1; \ldots; \texttt{begin var}\ x_m\colon m_m := t_m; s\ \texttt{end end}]\!] \end{aligned}$$

4.3.3 Prozeduren

Prozeduren mit Parametern: verwende *Umgebungen*, hier aber parameterlose Prozeduren: Menge P von Prozedurbezeichnern, dann:

(10) *Aufruf einer Prozedur*: Ist $p \in P$, so ist

$$\texttt{call}\ p \in \mathsf{STAT}$$

(11) *Deklaration einer Prozedur*: Ist $p \in P$, $s \in \mathsf{STAT}$ und $\texttt{call}\ p \notin s$, so ist

$$(p := \texttt{proc}\ (s)) \in \mathsf{STAT}$$

Ein Speicher ist nun

$$\sigma \colon X \cup P \to \left(\bigcup_{m \in S} i[m] \right) \cup \underbrace{\mathsf{PROC}}_{\mathsf{STORE}^{\mathsf{STORE}}}$$

Umsetzung mit direkter Summe:

$$\mathsf{STORE} = \left((X \oplus P) \to \left(\left(\bigoplus_{m \in S} i[m] \right) \oplus \mathsf{STORE}^{\mathsf{STORE}} \right) \right)^{\perp}$$

Dann Semantik-Festlegung:

(10) *Aufruf einer Prozedur*:

$$[\![\texttt{call}\ p]\!](\sigma) = \begin{cases} (\sigma(p))(\sigma) & \text{falls} \quad \sigma \neq \perp \\ \perp & \text{falls} \quad \sigma = \perp \end{cases}$$

(11) *Deklaration einer Prozedur*:

$$[\![p := \texttt{proc}\ (s)]\!](\sigma) = \begin{cases} \sigma[p \leq [\![s]\!]] & \text{falls} \quad \sigma \neq \perp \\ \perp & \text{falls} \quad \sigma = \perp \end{cases}$$

4.3.4 Seiteneffekte

Veränderung u.a.:

$$[\![t]\!] \colon \mathsf{STORE} \to (i[m] \times \mathsf{STORE})$$

Seiteneffekte sind aber böse :o).

5 Zur Lösung rekursiver Bereichsgleichungen

5.1 Adjungierte Paare und Retraktionsfolgen

$$D = \mathcal{E}(D)$$

5.1.1 Beispiele (Bereichsgleichungen)

Nichtrekursive Bereichsgleichung:

$$\mathsf{STATE} = \mathsf{IN} \otimes \mathsf{STORE} \otimes \mathsf{OUT}$$

Rekursive Bereichsgleichung:

$$\mathsf{STROKE} = \{\varepsilon\}^{\perp} \oplus \{|\}^{\perp} \otimes \mathsf{STROKE}$$

Lösung dieser Gleichung ist die CPO $(\{\perp\} \cup \{|\}^*, \leq)$, d.h. die Menge

$$\{\perp, \varepsilon, |, ||, |||, ||||, \ldots\}$$

Verallgemeinerung:

$$\mathsf{SEQU} = \{\varepsilon\}^{\perp} \oplus M^{\perp} \times \mathsf{SEQU}$$

Dies ergibt die CPO $((M^{\perp})^* \cup (M^{\perp})^{\infty}, \leq)$. ???

Weiteres Beispiel (frühes Pascal):

```
program self(input, output);
  function F(n: integer; g:function):integer;
    begin
      if n = 0 then F := 1
               else F := n * g(n - 1, g)
    end;
  end;
begin
  write(F(5, F))
end.
```

Auswertung:

$$F(5, F) \to 5 \cdot F(4, F) \to \ldots \to 120 \cdot F(0, F) \to 120$$

Frage: Typ des Parameters g, d.h. welche CPO $(M, \leq)$ für den Typ *function*?
Dies ergibt Bereichsgleichung

$$M = [\mathbb{N}^{\perp} \times M \to \mathbb{N}^{\perp}]$$

Lösung durch Retraktion aus einer universellen CPO (nicht hier) oder Konstruktion als *inverser Limes einer Retraktionsfolge von CPOs*.

5.1.2 Definition (Adjungiertes Paar, Einbettung, Projektion)

Seien $(M, \leq)$ und $(N, \leq)$ zwei CPOs. Sei ein Paar φ, ψ von Funktionen gegeben mit $\varphi\colon M \to N$ und $\psi\colon N \to M$. Das Paar (φ, ψ) heißt *adjungiertes Paar* von M nach N, falls gilt:

1. φ und ψ sind stetig
2. $\psi \circ \varphi = \mathrm{id}_M$ bzw. $\psi(\varphi)(u) = u \; \forall \, u \in M$
3. $\varphi \circ \psi \leq \mathrm{id}_N$ bzw. $\varphi(\psi)(u) = v \; \forall \, v \in N$

φ ist *Einbettung*, ψ *Projektion* und M das *Retrakt* von N mittels (φ, ψ).

$$(M, \leq) \preceq (N, \leq) \;:\Longleftrightarrow\; \exists\, (\varphi, \psi) \text{ adj. Paar von } (M, \leq) \text{ nach } (N, \leq)$$

5.1.3 Definition (Retraktionsfolge)

Eine *Retraktionsfolge* $(M_i, \varphi_i, \psi_i)_{i \geq 0}$ ist abzählbar unendliche Folge von CPOs $(M_i, \leq)_{i \leq 0}$ und entsprechende adjungierte Paare (φ_i, ψ_i):

$$M_0 \underset{\psi_0}{\overset{\varphi_0}{\rightleftarrows}} M_1 \underset{\psi_1}{\overset{\varphi_1}{\rightleftarrows}} M_2 \underset{\psi_2}{\overset{\varphi_2}{\rightleftarrows}} M_3 \rightleftarrows \ldots$$

5.1.4 Beispiel (Spezielle Retraktionsfolge)

Mögliche Retraktionsfolge bei natürlichen Zahlen (mit $\psi_i = \mathrm{id}[i+1 \leftarrow i]$):

$$\{0\} \underset{\psi_0}{\overset{\mathrm{id}}{\rightleftarrows}} \{0,1\} \underset{\psi_1}{\overset{\mathrm{id}}{\rightleftarrows}} \{0,1,2\} \underset{\psi_2}{\overset{\mathrm{id}}{\rightleftarrows}} \{0,1,2,3\} \rightleftarrows \ldots$$

5.1.5 Definition (Inverser Limes)

Sei $(M_i, \varphi_i, \psi_i)_{i \geq 0}$ Retraktionsfolge. *Inverser Limes*:

$$\varprojlim M_i := \left\{ \langle u_i \rangle_{i \geq 0} \in \prod\nolimits_{i \geq 0} M_i \;\middle|\; \forall\, i \geq 0\colon u_i = \psi_i(u_{i+1}) \right\}$$

Verallgemeinerte Produktordnung:

$$\langle u_i \rangle_{i \geq 0} \leq \langle v_i \rangle_{i \geq 0} \;:\Longleftrightarrow\; \forall i \geq 0\colon u_i \leq v_i$$

5.1.6 Satz (CPO-Eigenschaft inverser Limes)

Sei $(M_i, \varphi_i, \psi_i)_{i \geq 0}$ Retraktionsfolge. Dann ist $(\varprojlim M_i, \leq)$ CPO.

5.2 Die Retraktionsfolge zur Gleichung $D = [D \to D]$

5.2.1 Satz (Startpunkt)

Sei $(M_0, \leq)$ eine CPO (normalerweise mit mindestens zwei Elementen). Dann ist folgendes ein adjungiertes Paar:

$$M \underset{f \mapsto f(\bot)}{\overset{u \mapsto \bar{u}}{\rightleftarrows}} [M \to M]$$

5.2.2 Satz (Induktiver Aufbau)

Sei $(M, \leq)$ CPO und (φ, ψ) adjungiertes Paar von M zu $[M \to M]$. Dann existiert ein adjungiertes Paar (Φ, Ψ) von $[M \to M]$ zu $[[M \to M] \to [M \to M]]$ mit

$$\begin{array}{lcl} \Phi(f) & := & \varphi \circ f \circ \psi \colon [[M \to M] \to [M \to M]] \\ \Psi(g) & := & \psi \circ f \circ \varphi \colon [M \to M] \end{array}$$

5.2.3 Zusammenfassung: Retraktionsfolge zu $D = [D \to D]$

Aus 5.2.1 und 5.2.2 folgt die Existenz der Retraktionsfolge:

$$M_0 \underset{\psi_0}{\overset{\varphi_0}{\rightleftarrows}} M_1 \underset{\psi_1}{\overset{\varphi_1}{\rightleftarrows}} M_2 \underset{\psi_2}{\overset{\varphi_2}{\rightleftarrows}} M_3 \rightleftarrows \ldots$$

Diese hat folgende Eigenschaften:

(i) $\varphi_0(u) = \bar{u} \; \forall \; u \in M_0$

(ii) $\psi_0(g) = g(\bot) \; \forall \; g \in M_1$

(iii) $\varphi_{i+1}(f) = \varphi_i \circ f \circ \psi_i \; \forall \; f \in M_{i+1}$

(iv) $\psi_{i+1}(g) = \psi_i \circ g \circ \varphi_i \; \forall \; g \in M_{i+2}$

5.3 Kegelbildung

5.3.1 Definition

Sei $(M_i, \varphi_i, \psi_i)_{i \geq 0}$ Retraktionsfolge. Definiere abkürzend:

$$\begin{array}{lll} \varphi_i^j \colon M_i \to M_j & \text{mit} & \varphi_i^j = \varphi_{j-1} \circ \ldots \circ \varphi_i \\ \psi_i^j \colon M_j \to M_i & \text{mit} & \psi_i^j = \psi_i \circ \ldots \circ \psi_{j-1} \end{array}$$

5.3.2 Satz

Sei $(M_i, \varphi_i, \psi_i)_{i\geq 0}$ Retraktionsfolge. Dann ist (φ_i^j, ψ_i^j) ein adjungiertes Paar von M_i nach M_j.

5.3.3 Satz (Kegelbildung)

Sei $(M_i, \varphi_i, \psi_i)_{i\geq 0}$ Retraktionsfolge. Erweiterung zum Kegel:

$$\begin{array}{lll} \varphi_i^\infty \colon M_i \to \varprojlim M_i & \text{mit} & \varphi_i^\infty(u)_j = \begin{cases} \varphi_i^j(u) & \text{falls} \quad i \leq j \\ \psi_j^i(u) & \text{falls} \quad i > j \end{cases} \\ \psi_i^\infty \colon \varprojlim M_i \to M_i & \text{mit} & \psi_i^\infty(\langle u_k \rangle_{k\geq 0}) = u_i \end{array}$$

Dann ist:

$$\varphi_i^\infty(u) = \left\langle \psi_0^i(u), \ldots, \psi_{i-2}^i(u), \psi_{i-1}^i(u), u, \varphi_i^{i+1}(u), \varphi_i^{i+2}(u), \ldots \right\rangle$$

5.3.5 Satz

Sei $(M_i, \varphi_i, \psi_i)_{i\geq 0}$ Retraktionsfolge und sei $\langle u_i \rangle_{i\geq 0} \varprojlim M_i$. Dann bilden die folgenden Familien jeweils Ketten:

$$\langle \varphi_i^\infty(u_i) \rangle_{i\geq 0} \quad \text{und} \quad \langle \varphi_n^\infty \circ \psi_n^\infty \rangle_{n\geq 0}$$

Dann gilt:

$$\bigsqcup_{i\geq 0} \varphi_i^\infty(u_i) = \langle u_i \rangle_{i\geq 0} \quad \text{und} \quad \bigsqcup_{n\geq 0} \varphi_n^\infty \circ \psi_n^\infty = \mathrm{id}_{\varprojlim M_i}$$

5.4 Lösung der Gleichung $D = [D \to D]$

5.4.1 Definition (Isomorphismus zu $D = [D \to D]$)

Sei $(M_i, \varphi_i, \psi_i)_{i\geq 0}$ Retraktionsfolge aus 5.2, die 5.2.3 erfüllt.

$$\begin{array}{lll} \Phi \colon \varprojlim M_i \to [\varprojlim M_i \to \varprojlim M_i] & \text{mit} & \Phi \colon \langle u_i \rangle_{i\geq 0} \mapsto \bigsqcup_{n\geq 0} \varphi_n^\infty \circ u_{n+1} \circ \psi_n^\infty \\ \Psi \colon [\varprojlim M_i \to \varprojlim M_i] \to \varprojlim M_i & \text{mit} & \Psi \colon f \mapsto \bigsqcup_{n\geq 0} \varphi_{n+1}^\infty(\psi_n^\infty \circ f \circ \varphi_n^\infty) \end{array}$$

5.4.2 Satz (Dana Scott, 1969)

Sei $(M_i, \varphi_i, \psi_i)_{i\geq 0}$ Retraktionsfolge aus 5.2, die 5.2.3 erfüllt. Dann gilt:

a) Φ und Ψ sind stetig

b) Φ und Ψ sind invers:

(1) $\Psi \circ \Phi = \mathrm{id}_{\underset{\longleftarrow}{\mathbf{lim}}\, M_i}$

(2) $\Phi \circ \Psi = \mathrm{id}_{\left[\underset{\longleftarrow}{\mathbf{lim}}\, M_i \to \underset{\longleftarrow}{\mathbf{lim}}\, M_i\right]}$

Damit sind die beiden CPOs $\underset{\longleftarrow}{\mathbf{lim}}\, M_i$ und $[\underset{\longleftarrow}{\mathbf{lim}}\, M_i \to \underset{\longleftarrow}{\mathbf{lim}}\, M_i]$ **isomorph** als CPOs!

6 Programmverifikation und axiomatische Semantik

6.1 Motivation

Situation bei Verifikation für $\mathcal{L}_A$:

gegebenes Programm	$t \in \mathsf{EXP}_m$
gegebene Spezifikation	$u \in i[m]$
gegebene Zuordnung	$a \in \mathsf{ENV}$
zu zeigen ist	$[\![t]\!](a) = u$

Situation bei Verifikation für $\mathcal{L}_P$:

gegebenes Programm	$t \in \mathsf{STAT}$
gewünschter Speicherzustand	$\sigma' \in \mathsf{STORE}$
gegebener Speicherzustand	$\sigma \in \mathsf{STORE}$
zu zeigen ist	$[\![t]\!](\sigma) = \sigma'$

6.2 Verifikation mittels Fixpunkttheorie

Methoden:

(1) Berechnungsinduktion (siehe 2.3.7)

(2) Lemma von Park (siehe 2.3.11)

(3) direkte Ausnutzung der Eigenschaften eines kleinsten Fixpunktes

6.3 Korrektheitsbegriffe bei prozeduralen Sprachen

6.4 Der Hoare-Kalkül

6.4.1 Syntax der Hoare-Logik

Seien Σ_{ASS} und X Signatur bzw. Programmvariablenmenge von $\mathcal{L}_P$. Formeln im Hoare-Kalkül:

(1) Die Menge ASS der Zusicherungen.

(2) Die Menge HF der Formeln der Gestalt $\{\varphi\}\ s\ \{\psi\}$ mit $\varphi, \psi \in \mathsf{ASS}$ und $s \in \mathsf{STAT}$.

6.4.2 Semantik der Hoare-Logik (Gültigkeit)

Eine Hoare-Formel $\{\varphi\}\ s\ \{\psi\} \in \mathsf{HF}$ ist *gültig* ($\models \{\varphi\}\ s\ \{\psi\}$), falls das Programm s partiell korrekt bezüglich der Vorbedingung φ und der Nachbedingung ψ ist:

$$\models \{\varphi\}\ s\ \{\psi\} \ :\Longleftrightarrow\ \forall\, \sigma \in \mathsf{STORE} \setminus \{\bot\}\colon \left\{ \begin{array}{ll} & \models \varphi[\sigma] \\ \wedge & [\![s]\!](\sigma) \neq \bot \Rightarrow \models \psi[[\![s]\!](\sigma)] \end{array} \right.$$

6.4.3 Kalküle

Ein Kalkül besteht aus

a) Menge $\mathcal{A} \subseteq \mathcal{A}$ von *Axiomen*

b) Menge von *Herleitungsregeln* der Form

$$\frac{F_1 \ldots F_k}{F}$$

Die Menge $\mathcal{H}$ der Herleitungen mit Wurzelfunktion

$$\mathrm{root}\colon \mathcal{H} \to \mathcal{F}$$

a) Für alle $F \in \mathcal{A}$ gilt $F \in \mathcal{H}$ und $\mathrm{root}(F) = F$.

b) Für alle Herleitungsregeln

$$\frac{F_1 \ldots F_k}{F}$$

und alle $\mathcal{H}_i \in \mathcal{H}$ mit $\mathrm{root}(\mathcal{H}_i) = F_i$ gilt

$$\frac{\mathcal{H}_1 \ldots \mathcal{H}_k}{F} \in \mathcal{H}$$

$F \in \mathcal{F}$ ist herleitbar ($\models F$) genau dann, wenn $H \in \mathcal{H}$ mit $\mathrm{root}(H) = F$ existiert.

- Ein Kalkül ist *vollständig*, falls $\models F \Rightarrow \models F$.
- Ein Kalkül ist *korrekt*, falls $\models F \Rightarrow \models F$

6.4.4 Hoare-Kalkül

Axiome des Hoare-Kalküls:

(1) Für alle $\varphi \in \mathsf{ASS}$ mit $\vdash \varphi$ ist φ ein Axiom.

(2) *Axiom der leeren Anweisung*:

$$\{\varphi\}\ \texttt{skip}\ \{\varphi\}$$

(3) *Axiom der undefinierten Anweisung*:

$$\{\varphi\}\ \texttt{abort}\ \{\psi\}$$

(4) *Zuweisungsaxiom*:

$$\{\varphi[t \leftarrow x]\}\ x := t\ \{\varphi\}$$

Herleitungsregeln:

(5) *Erste Konsequenzregel* (Abschwächung der Vorbedingung)

$$\frac{\varphi_1 \to \varphi_2 \quad \{\varphi_2\}\ s\ \{\psi\}}{\{\varphi_1\}\ s\ \{\psi\}}$$

(6) *Zweite Konsequenzregel* (Verstärkung der Nachbedingung)

$$\frac{\psi_1 \to \psi_2 \quad \{\varphi\}\ s\ \{\psi_1\}}{\{\varphi_1\}\ s\ \{\psi_2\}}$$

(7) *Regel der Alternative*

$$\frac{\{\varphi \wedge b\}\ s_1\ \{\psi\} \quad \{\varphi \wedge \mathrm{not}(b)\}\ s_2\ \{\psi\}}{\{\varphi\}\ \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{fi}\ \{\psi_2\}}$$

(8) *Regel der while-Schleife*

$$\frac{\{I \wedge b\}\ s\ \{I\}}{\{I\}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{od}\ \{I \wedge \mathrm{not}(b)\}}$$

(9) *Regel der Komposition*

$$\frac{\{\varphi\}\ s_1\ \{\varrho\} \quad \{\varrho\}\ s_2\ \{\psi\}}{\{\varphi\}\ s_1; s_2\ \{\psi\}}$$

Nun:

$$\vdash \{\varphi\}\ s\ \{\psi\} \ :\Longleftrightarrow\ \{\varphi\}\ s\ \{\psi\} \text{ mittels (1) bis (9) herleitbar}$$

6.4.6 Satz

Gegeben seien $\varphi, \psi \in \mathsf{ASS}$ und $s \in STAT$. Gibt es ein $I \in \mathsf{ASS}$ mit den folgenden Eigenschaften

$$(1) \vdash I \wedge \mathrm{not}(b) \to \psi \quad (2) \vdash \{\varphi\}\ s_1\ \{I\} \quad (3) \vdash \{I \wedge b\}\ s_2\ \{I\}$$

Dann gilt

$$\vdash \{\varphi\}\ s_1; \texttt{while}\ b\ \texttt{do}\ s_2\ \texttt{od}\ \{\psi\}$$

6.4.9 Satz (Gültigkeit der Axiome)

Die Axiome (1) bis (4) des Hoare-Kalküls sind gültig.

6.4.10 Satz (Konsequenzregeln erhalten Gültigkeit)

Die Regeln (5) und (6) des Hoare-Kalküls erhalten die Gültigkeit.

6.4.11 Satz (Regeln der Alternative und der Komposition erhalten Gültigkeit)

Die Regeln (7) und (9) des Hoare-Kalküls erhalten die Gültigkeit.

6.4.12 Satz (While-Regel erhält Gültigkeit)

Die Regel (8) des Hoare-Kalküls erhält die Gültigkeit.

6.4.13 Satz (Korrektheit des Hoare-Kalküls)

Für alle $\{\varphi\}\ s\ \{\psi\} \in \mathsf{HF}$ gilt:

$$\vdash \{\varphi\}\ s\ \{\psi\} \Longrightarrow \vdash \{\varphi\}\ s\ \{\psi\}$$