Semantik und Programmverifikation

Prof. Dr. Christoph Walther / Simon Siegler
Technische Universität Darmstadt — Wintersemester 2008/09

Hausaufgabe mit Lösungsvorschlag 5

Hausaufgabe 5.1 (Datenstrukturen und operationale Semantik) **(2 Punkte)**

Gegeben sei das funktionale Programm $P = \langle D_{list}, F_{outer} \rangle$ mit $D_{list} =$

$$\textbf{structure } \epsilon, \quad add(head : nat, tail : list) : list$$

und F_{outer} wie üblich. Bestimmen Sie

$$\mathsf{sem}_{\mathrm{op}}\,[\![P]\!]\,(eq_{list}(tail(add(0,\epsilon)), tail(add(outer(0),\epsilon))))$$

und vergleichen Sie dieses Resultat mit dem Ergebnis, das man erhält, wenn Regel (3) des Auswertungskalküls aus Definition 2.5.4 für beliebige $q_1, \ldots, g_{n_h} \in \mathcal{T}(\Sigma(P))_{s_{h,1},\ldots,s_{h,n_h}}$ anwendbar ist.

Lösungsvorschlag:

$$\begin{aligned}
& eq_{list}(tail(add(0,\epsilon)), tail(add(outer(0),\epsilon))) \\
\to_P \quad & eq_{list}(\epsilon, tail(add(outer(0),\epsilon))) \\
\to_P \quad & eq_{list}(\epsilon, tail(add(succ(outer(0)),\epsilon))) \\
\to_P \quad & \ldots \\
\Rightarrow \quad & \mathsf{sem}_{\mathrm{op}}\,[\![P]\!]\,(eq_{list}(tail(add(0,\epsilon)), tail(add(outer(0),\epsilon)))) = \propto
\end{aligned}$$

Mit geänderter Regel (3) erhält man

$$\begin{aligned}
& eq_{list}(tail(add(0,\epsilon)), tail(add(outer(0),\epsilon))) \\
\to_P \quad & eq_{list}(\epsilon, tail(add(outer(0),\epsilon))) \\
\to_P \quad & eq_{list}(\epsilon, \epsilon) \\
\to_P \quad & true \\
\Rightarrow \quad & \mathsf{sem}_{\mathrm{op}}\,[\![P]\!]\,(eq_{list}(tail(add(0,\epsilon)), tail(add(outer(0),\epsilon)))) = true
\end{aligned}$$

Hausaufgabe 5.2 (Datenstrukturen) **(5 Punkte)**

(a) Zeigen Sie, dass Satz 2.2.2 auch für funktionale Programme mit Datenstrukturen gilt.

Lösungsvorschlag:

(i) $t \in \min_{\to_P}(\mathcal{T}(\Sigma(P))) \Leftrightarrow t \in \mathcal{T}(\Sigma^c)$
Beweis durch strukturelle Induktion über t

Schrittfall f Konstruktorsymbol: analog zu *succ*
Schrittfall f Selektorsymbol: analog zu *pred*
sonst die anderen Fälle sind wie bei Satz 2.2.2.(i)

(ii) $\forall t, s_1, s_2 \in \mathcal{T}(\Sigma(P))$ mit $t \to_P s_1 \wedge t \to_P s_2$ gilt $s_1 = s_2$ oder $\exists r \in \mathcal{T}(\Sigma(P)).s_1 \to_P r \wedge s_2 \to_P r$
Beweis durch strukturelle Indukion über t

Schrittfall f Konstruktorsymbol: analog zu *succ*
Schrittfall f Selektorsymbol: analog zu *pred*
sonst die anderen Fälle sind wie bei Satz 2.2.2.(ii)

(iii) $\rightarrow_P$ konfluent
Beweis folgt wie in Satz 2.2.1.(iii) und Satz 2.2.2.(iii) aus (ii)

(iv) Für alle Auswertungsfolgen $\langle t_i \rangle_{i \in J}$ und $\langle s_i \rangle_{i \in K}$ gilt $|\langle t_i \rangle_{i \in J}|_P = |\langle s_i \rangle_{i \in K}|_P$.
Beweis: exakt analog zu 2.2.2.(iv)

(v) Für jedes Programm $P \in \mathcal{FP}$ und jeden Term $t \in \mathcal{T}(\Sigma(P))$ existiert *höchstens ein* Term $q \in \mathcal{T}(\Sigma^c)$ mit $t \rightarrow_P^* q$.
Beweis: Analog zu 2.2.2.(v): Zeigt man mit (i) und (iii) wie in Satz 2.2.1.(v)

(b) Sei P ein funktionales Programm mit Datenstrukturen, das keine Funktionsprozeduren enthält. Beweisen Sie, daß $\rightarrow_P$ fundiert ist.

Lösungsvorschlag:

Idee: finde eine Massfunktion $|\cdot| : \mathcal{T}(\Sigma, \mathcal{V}) \rightarrow \mathbb{N}$ so, dass Satz 1.4.1.(ii) anwendbar wird. Dazu ist für jede der $\frac{t}{r}$ Regeln (1) bis (11) aus Definition 2.5.4 zu zeigen, dass $|t| >_{\mathbb{N}} |r|$. (Da P keine Funktionsprozeduren enthält, braucht Regel (12) nicht betrachtet zu werden.)

Vermutung: Anzahl der Selektoren im Term, wobei auch if und eq mitgezählt werden, also:

$$\begin{array}{lrcll} (T1) & |f(t_1, \ldots, t_n)| & = & \sum_{i=1}^n |t_i| & \text{falls } f \in \Sigma^c \\ (T2) & |f(t_1, \ldots, t_n)| & = & 1 + \sum_{i=1}^n |t_i| & \text{falls } f \in \Sigma^d \end{array}$$

Zeige: $|t| > |r|$, falls $t \rightarrow_P r$ und t ein Term ohne Funktionsprozedursymbole.

Beweis durch strukturelle Induktion über t:

Basisfall t Konstante oder Variable: es ist nichts zu zeigen, weil der Auswertungskalkül keine anwendbare Regel enthält und folglich kein r mit $t \rightarrow_P r$ existiert.

Schrittfall $t = f(t_1, \ldots, t_n)$: Wir gehen in der Folge davon aus, dass $t \notin \min_{\rightarrow_P}(\mathcal{T}(\Sigma(P)))$, weil sonst nichts zu zeigen ist (M).

Fall f ist Konstruktorsymbol: Es gibt wegen (M) einen unmittelbaren Teilterm t_i mit $t_i \notin \min_{\rightarrow_P}(\mathcal{T}(\Sigma(P))) \Rightarrow$ es gibt einen Term s mit $t_i \rightarrow_P s \Rightarrow$ genau Regel (1) anwendbar mit (T1) $\Rightarrow |t| = \sum_{i=1} t_i, |r| = |s| - |t_i| + |t| \Rightarrow$ mit IH $|t_i| > |s|$ folgt $|t| > |r|$

Fall f ist Selektorsymbol: $\Rightarrow t = f(t_1)$ für ein $t_1 = g(s_1, \ldots, s_n)$

Fall $g \in \Sigma^c$:

Selektor f passt zu Konstruktor g: $\Rightarrow$ nur Regel (3) anwendbar:
$|t| = 1 + \sum_{i=1}^n s_i$, $|r| = s_j$ für ein $j \in \{1, \ldots, n\} \Rightarrow |t| > |r|$

Selektor f passt nicht zu Konstruktor g: $\Rightarrow$ nur Regel (2) anwendbar
$|t| = 1 + \sum_{i=1}^n s_i$, aber $|r| = 0$, denn Beispielterme sind Konstruktorgrundterme $\Rightarrow |t| > |r|$.

Fall $g \notin \Sigma^c$: $\Rightarrow$ mit (M) ist Regel (4) die einzig anwendbare und es gibt einen Term s mit $t_1 \rightarrow_P s \Rightarrow$ mit IH ist $|t_1| > |s| \Rightarrow |t| = 1 + |t_1| > |r| = 1 + |s|$

Fall $f = eq_s$:

Fall $t_1, t_2 \in \mathcal{T}(\Sigma^c)_s$:

Fall $t_1 = t_2$: $\Rightarrow$ nur Regel (5) anwendbar: $|t| = 1 + |t_1| + |t_2| = 1$, $|r| = 0$, $\Rightarrow |t| > |r|$

Fall $t_1 \neq t_2$: analog für Regel (6)

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_s$: $\Rightarrow t_1 \notin \min_{\rightarrow_P}(\mathcal{T}(\Sigma(P))) \Rightarrow$ es gibt einen Term s mit $t_1 \rightarrow_P s \Rightarrow$ mit IH $|t_1| > |s|$
nur Regel (7) anwendbar $\Rightarrow |t| = 1 + |t_1| + |t_2| > |r| = 1 + |s| + |t_2|$

Fall $t_2 \notin \mathcal{T}(\Sigma^c)_s$**:** analog mit Regel (8)

Fall $f = if_s$**:**

Fall $t_1 \in \mathcal{T}(\Sigma^c)_s$**:**

Fall $t_1 = true$**:** $\Rightarrow$ nur Regel (9) anwendbar
$|t| = 1 + 1 + |t_2| + |t_3|$, $|r| = |t_2| \Rightarrow |t| > |r|$

Fall $t_1 = false$**:** analog für Regel (10)

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_s$**:** $\Rightarrow t_1 \notin \min_{\rightarrow_P}(\mathcal{T}(\Sigma(P)))$ es gibt einen Term s mit $t_1 \rightarrow_P s \Rightarrow$ Regel (11) ist einzig anwendbare
per IH folgt $|t_1| > |s| \Rightarrow |t| = 1 + |t_1| + |t_2| + |t_3| > |r| = 1 + |s| + |t_2| + |t_3|$

Anmerkung zum Fall f **ist definiertes Funktionssymbol:** in diesem Fall wäre Regel (12) anzuwenden, in der der Funktionsaufruf durch den instanziierten Rumpf zu ersätzen wäre. Der Rumpf ist im allgemeinen deutlich komplexer als der Aufruf, so dass die Gefahr besteht, dass hier die Bedingung $|t| > |r|$ verletzt würde. Glücklicherweise schliesst die Aufgabestellung dies aus.

(vgl. Übung 2.5.2)

Hausaufgabe 5.3 (Parameterübergabeverfahren) **(5 Punkte)**

$P \in \mathcal{FP}$ sei ein funktionales Programm und $t, t_1, \ldots, t_n, r$ Terme aus $\mathcal{T}(\Sigma(P))$. Zeigen Sie:

(a) $\mathsf{cbv\text{-}eval}_P(t) = \mathsf{cbv\text{-}eval}_P(r)$, falls $t \Longrightarrow_P^* r$.

Lösungsvorschlag:

Fall $\mathsf{cbv\text{-}eval}_P(r) = \propto$**:** Annahme: $\mathsf{cbv\text{-}eval}_P(t) \neq \propto$, d. h. $\exists q \in \mathcal{T}(\Sigma^c(P)).t \Rightarrow_P^* q$, mit Konfluenz und $\Rightarrow_P$-Minimalität folgt $r \Rightarrow_P^* q \Rightarrow \mathsf{cbv\text{-}eval}_P(r) = q \not\downarrow \Rightarrow$ Annahme falsch $\Rightarrow \mathsf{cbv\text{-}eval}_P(t) = \propto$

Fall $\mathsf{cbv\text{-}eval}_P(r) = q$ **mit** $q \in \mathcal{T}(\Sigma^c(P))$**:**
$\Rightarrow r \Rightarrow_P^*$ und nach Vor: $t \Rightarrow_P^* r \Rightarrow t \Rightarrow_P^* q \Rightarrow \mathsf{cbv\text{-}eval}_P(t) = q$

(b) $\mathsf{cbv\text{-}eval}_P(f(t_1, \ldots, t_i, \ldots, t_n)) = \mathsf{cbv\text{-}eval}_P(f(t_1, \ldots, \mathsf{cbv\text{-}eval}_P(t_i), \ldots, t_n))$,
falls $\mathsf{cbv\text{-}eval}_P(f(t_1, \ldots, t_i, \ldots, t_n)) \neq \propto$ und $i = 1$ für $f = if_s$.

Lösungsvorschlag:

Für die Fälle $f \in \Sigma(BM)$ funktioniert der Beweis genau so wie für eval_P. Daher betrachten wir nur $f \in \Sigma(P) \setminus \Sigma(BM)$.

Es gilt: $\mathsf{cbv\text{-}eval}_P(f(t_1, \ldots, t_n)) \neq \propto$. Annahme: $\exists i \in \{1, \ldots, n\}$ mit $\mathsf{cbv\text{-}eval}_P(t_i) = \propto \Rightarrow$ es gibt eine unendliche Auswertungsfolge $\langle r_j \rangle_{j \in \mathbb{N}}$ für $t_i \Rightarrow \langle f(t_1, \ldots, r_j, \ldots, t_n) \rangle_{j \in \mathbb{N}}$ eine unendliche Auswertungsfolge für $f(t_1, \ldots, t_n) \Rightarrow \mathsf{cbv\text{-}eval}_P(f(t_1, \ldots, r_j, \ldots, t_n)) = \propto \not\downarrow$.

Also gilt: $\forall i \in \{1, \ldots, n\}.\mathsf{cbv\text{-}eval}_P(t_i) \neq \propto$.

Sei jetzt $i \in \{1, \ldots, n\}$ beliebig aber fest $\Rightarrow \mathsf{cbv\text{-}eval}_P(t_i) \neq \propto \Rightarrow$ es gibt eine endliche Auswertungsfolge $\langle r_j \rangle_{j \leq n_o}$ für t_i $f(t_1, \ldots, r_0, \ldots, t_n) \Rightarrow_P^* f(t_1, \ldots, r_{n_0}, \ldots, t_n)$ und da $r_{n_0} \in \mathcal{T}(\Sigma^c(P))$ $\Rightarrow f(t_1, \ldots, r_0, \ldots, t_n) \Rightarrow_P^* f(t_1, \ldots, \mathsf{cbv\text{-}eval}_P(r_0), \ldots, t_n)$ und mit (a) folgt die Behauptung.