Semantik und Programmverifikation

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

Hausaufgabe mit Lösungsvorschlag 2

Hausaufgabe 2.1 (Auswertungskalkül der Basismaschine) **(5 Punkte)**

Nehmen wir an, der Auswertungskalkül der Basismaschine (vergleiche Definition 2.2.1) wird um die folgenden zwei Ableitungsregeln erweitert:

$$(A)\ \frac{if_s(b,p_1,p_2)}{if_s(b,p_1',p_2)},\ \text{falls}\ p_1 \to_{BM} p_1'; \qquad (B)\ \frac{if_s(b,p_1,p_2)}{if_s(b,p_1,p_2')},\ \text{falls}\ p_2 \to_{BM} p_2'.$$

Untersuchen Sie die Konsequenzen dieser Erweiterung für $\to_{BM}$ und $\to_P$. Prüfen Sie dabei insbesondere, welche Aussagen aus den Sätzen 2.2.1 und 2.2.2 jetzt nicht mehr gültig sind (argumentativ, ohne formalen Beweis). Welche Auswirkungen besitzt diese Erweiterung auf die Implementierung von eval_P und eval_{BM}?

Lösungsvorschlag:

zu Satz 2.2.1:

(i) Die Aussage gilt weiterhin. Der Induktionsbeweis kann im Induktionsschritt erweitert werden. Für die beiden neuen Fälle kann der Beweis zum Fall $t = if_s(b,p_1,p_2)$ in Verbindung mit Regel (11) aus Definition 2.2.1 direkt übertragen werden (statt b betrachtet man p_1 bzw. p_2).

(ii) Diese Aussage ist nicht mehr gültig. Gegenbeispiel:

$$if_s(\text{TRUE},0,0) \leftarrow_{BM} if_s(\text{TRUE},0,pred(0)) \to_{BM} 0$$

und weder $if_s(\text{TRUE},0,0) =_{\mathcal{T}} 0$ noch existiert ein q mit $if_s(\text{TRUE},0,0) \to_{BM} q \leftarrow_{BM} 0$, denn 0 ist bereits minimal.

(iii) Obwohl (ii) nicht mehr gewährleistet ist, gilt trotzdem noch die strenge Konfluenz. (Die Aussage in (ii) ist eine Verschärfung der strengen Konfluenz.)

Dies kann man leicht zeigen, indem man den Beweis zu (ii) des Satzes 2.2.1 verwendet und im Induktionsschritt um 2 Fälle (aufgrund der 2 neuen Regeln) erweitert. Wegen der strengen Konfluenz ist dann auch die Konfluenz gesichert.

(iv) Die Aussage bleibt erhalten. Die im Beweis von Satz 2.2.1.(iv) aufgestellte Behauptung (*) gilt weiterhin. Der Induktionsbeweis kann leicht um die 2 hinzukommenen Fälle erweitert werden.

(v) Auch diese Aussage bleibt weiterhin gültig. Der Beweis hierzu basiert auf den Aussagen (i), (iii) und (iv), die wie gesehen auch im erweiterten Kalkül gelten. Der Beweis kann damit unverändert übernommen werden

zu Satz 2.2.2:

(i) Da Satz 2.2.1.(i) auch für den erweiterten Kalkül der Basismaschine gültig bleibt, gilt auch diese Aussage weiterhin. Der Beweis dazu bleibt unverändert.

(ii) Diese Aussage gilt nicht mehr. Wenn sie schon für die Basismaschine nicht mehr richtig ist, dann natürlich erst recht nicht für den Kalkül für P.

(iii) Diese Aussage bleibt bestehen. Wie auch bei der Betrachtung zu Satz 2.2.1 ist Aussage (ii) nicht mehr gültig, die strenge Konfluenz aber weiterhin gewährleistet, denn aus der strengen Konfluenz von $\to_{BM}$ (Satz 2.2.1.(iii)) und der Aussage (ii) in Satz 2.2.2 folgt unmittelbar die strenge Konfluenz von $\to_P$. Diese reicht wiederum für die Konfluenz von $\to_P$ aus.

(iv) Diese Aussage gilt nicht mehr: $if_s(\text{TRUE}, 0, pred(0)) \to_P 0$ (1 Schritt), aber $if_s(\text{TRUE}, 0, pred(0)) \to_P if_s(\text{TRUE}, 0, 0) \to_P 0$ (2 Schritte).

(v) Der Beweis zu Satz 2.2.2.(v) kann unverändert übernommen werden, da die Aussage (i) und (iii) wie gesehen weiterhin gültig sind.

Zu den Auswirkungen auf eval_{BM} und $eval_P$ können wir folgendes feststellen:

Bislang existiert im Kalkül zu BM ein Indeterminismus: mit $t_1 \to_{BM} t_1'$ und $t_2 \to_{BM} t_2'$ können wir $eq_{nat}(t_1, t_2) \to_{BM} eq_{nat}(t_1', t_2)$ oder $eq_{nat}(t_1, t_2) \to_{BM} eq_{nat}(t_1, t_2')$ bilden.

Nun besitzt der Kalkül einen zweiten Indeterminismus bei if_s. Für eval_{BM} ist dies mit den Teilaussagen 2.2.1.(iii)+(iv) unproblematisch. Eventuell entstehen nun längere Auswertungen, wenn wir nicht relevante Teilterme auswerten, doch sind alle Auswertungen endlich und wir erhalten stets für ein t den gleichen Term $\mathsf{eval}_{BM}(t)$.

Bei eval_P ist die Situation anders. Nach wie vor ist die Konfluenz gesichert. Damit können zwei verschiedene Auswertungsfolgen nicht zu unterschiedlichen Konstruktorgrundtermen q_1, q_2 führen. Wir können nun allerdings unendliche Auswertungen konstruieren, obwohl es auch endliche Auswertungen gibt: $if_s(true, 0, inner(0))$ beispielsweise führt immer zu 0 — wenn die Auswertung endet. Doch wir können hierzu auch nicht endende Auswertungen bilden.

Hausaufgabe 2.2 (Auswertungsrelation für Programme) **(3 Punkte)**

Zeigen Sie, dass für ein funktionales Programm $P \in \mathcal{FP}$ und zwei Terme $t, r \in \mathcal{T}(\Sigma(P))$ mit $\mathsf{eval}_P(t) \neq \propto$ und $t \to_P^+ r$ die Aussage $|t|_P > |r|_P$ gilt.

Lösungsvorschlag:

zu zeigen: $\forall t, r \in \mathcal{T}(\Sigma(P))$ mit $\mathsf{eval}_P(t) \neq \propto$ und $t \to_P^+ r$ gilt: $|t|_P > |r|_P$.

Beweis:

Lemma $\mathsf{eval}_P(r) \neq \propto$: $\mathsf{eval}_P(t) \neq \propto \Rightarrow \exists q \in \mathcal{T}(\Sigma^c)$ mit $t \to_P^* q$. Da $\to_P$ konfluent (Satz 2.2.2.(iii)) und mit Voraussetzung $t \to_P^* r$ folgt: $\exists q' \in \mathcal{T}(\Sigma(P))$ mit $r \to_P^* q'$ und $q' \to_P^* q$. Da $q \in \mathcal{T}(\Sigma^c)$ folgt q ist $\to_P$-minimal. $\Rightarrow q' = q \Rightarrow r \to_P^* q \Rightarrow \mathsf{eval}_P(r) = q \neq \propto$.

Wir haben also $\mathsf{eval}_P(t) = q$ und $\mathsf{eval}_P(r) = q \Rightarrow |t|_P \neq \infty$ und $|r|_P \neq \infty$.

Weiter folgt, dass Auswertungsfolgen $\langle t_i \rangle_{i \leq j_0}$ und $\langle r_i \rangle_{i \leq j_1}$ existieren mit

$$\begin{array}{l} t_0 = t,\, t_{j_0} = q, \forall i \in \{0, \ldots, j_0{-}1\}.\, t_i \to_P t_{i+1} \Rightarrow |t|_P = |\langle t_i \rangle_{i \leq j_0}| = j_0{+}1 \\ r_0 = r,\, r_{j_1} = q, \forall i \in \{0, \ldots, j_1{-}1\}.\, r_i \to_P r_{i+1} \Rightarrow |r|_P = |\langle r_i \rangle_{i \leq j_1}| = j_1{+}1 \end{array}$$

Wir müssen also zeigen $j_0 + 1 > j_1 + 1$.

Nach Voraussetzung gilt $t \to_P^+ r$. Es gibt also eine endliche Folge $\langle s_i \rangle_{i \leq j_2}$ mit $s_0 = t$, $s_{j_2} = r$ und $\forall i \in \{0, \ldots, j_2 - 1\}. s_i \to_P s_{i+1}$.

Wir konstruieren jetzt eine *neue* Auswertungsfolge für t: $\bar{t}_i = \begin{cases} s_i, & \text{falls } i \leq j_2 \\ r_{i-j_2} & \text{falls } j_2 < i \leq j_2 + j_1 \end{cases}$

zu zeigen bleibt: $\langle \bar{t} \rangle_{i \leq j_2 + j_1}$ ist eine endliche Auswertungsfolge für t. Also

zeige $t = \bar{t}_0$**:** $\bar{t}_0 = s_0 = t$

zeige $q = \bar{t}_{j_2+j_1+1}$**:** $\bar{t}_{j_2+j_1+1} = r_{j_1+1} = q$

zeige $\forall i \in \{0, \ldots, j_2 + j_1\} \bar{t}_i \to_P \bar{t}_{i+1}$**:**

Fall $0 \leq i < j_2$**:** $\Rightarrow \bar{t}_i = s_i, \bar{t}_{i+1} = s_{i+1} \Rightarrow \bar{t}_i \to_P \bar{t}_{i+1}$

Fall $i = j_2$**:** $\Rightarrow \bar{t}_i = s_{j_2} = r = r_0, \bar{t}_{i+1} = r_1 \Rightarrow \bar{t}_i \to_P \bar{t}_{i+1}$, da $r_0 \to_P r_1$.

Fall $j_2 < i \leq j_2 + j_1$**:** $\Rightarrow \bar{t}_i = r_{i-j_2}, \bar{t}_{i+1} = r_{i+1-j_2} \Rightarrow \bar{t}_i \to_P \bar{t}_{i+1}$

$\Rightarrow \langle \bar{t} \rangle_{i \leq j_2 + j_1}$ ist eine endliche Auswertungsfolge für t.

$$\begin{array}{rrrcl} \Rightarrow & |\langle \bar{t} \rangle_{i \leq j_2+j_1}| & = & |\langle t_i \rangle_{i \leq j_0}| \\ \Leftrightarrow & j_2 + j_1 + 1 & = & j_0 + 1 \\ \Leftrightarrow & j_2 + j_1 & = & j_0 \\ (\text{da } j_2 \geq 1) \Rightarrow & j_1 & < & j_0 \\ \Leftrightarrow & j_1 + 1 < j_0 + 1 & & \\ \Rightarrow & |r|_P & < & |t|_P \end{array}$$

Hausaufgabe 2.3 (Auswertungsrelation für Programme) **(3 + 1 + 2 = 6 Punkte)**

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

(a) Zeigen Sie: $|f(t_1, \ldots, t_i, \ldots, t_n)|_P > |t_i|_P$, falls $\mathsf{eval}_P(f(t_1, \ldots, t_n)) \neq \propto$ und entweder $f \in \{pred, eq_{nat}\}$ oder ($f = if_s$ und $i = 1$).

Hinweis: Verwenden Sie Aufgabe 3.4 und Satz 2.2.2(iv).

Lösungsvorschlag:

Satz 2.2.2.(iv) sichert uns zu, dass alle Auswertungsfolgen zu einem Term t die gleiche Länge besitzen. Es ist damit ausreichend, *eine* Auswertungsfolge zu t zu konstruieren, um eine Aussage zu der Länge *aller* Auswertungsfolgen zu t machen zu können.

Fall $f(t) = pred(t)$**:**

Fall $t = 0$**:** $|pred(0)|_P = 2, |0|_P = 1$

Fall $t = succ(q), q \in \mathcal{T}(\Sigma^c)_{nat}$**:** $|pred(succ(q))|_P = 2, |succ(q)|_P = 1$

Fall $t \notin \mathcal{T}(\Sigma^c)_{nat}$**:** $\Rightarrow t$ ist nach Satz 2.2.2.(i) nicht $\to_P$-minimal und $\mathsf{eval}_P(t) \neq \propto$ (sonst wäre auch $\mathsf{eval}_P(pred(t)) = \propto$). $\Rightarrow \exists q \in \mathcal{T}(\Sigma^c)_{nat}$ mit $t \to_P^+ q$ und q ist $\to_P$-minimal. Damit existiert dann aber auch die Auswertung $pred(t) \to_P^+ pred(q) \to_P 0$, falls $q = 0$ oder die Auswertung $pred(t) \to_P pred(q) \to_P \hat{q}$, falls $q = succ(\hat{q})$, wobei $t \to_P^+ q$ und $pred(t) \to_P^+ pred(q)$ die gleiche Länge besitzen. Damit erhalten wir $|pred(t)|_P = |t|_P + 1$, also $|pred(t)|_P > |t|_P$.

Fall $f(t_1, t_2, t_3) = if_s(t_1, t_2, t_3)$ **und** $i = 1$**:**

Fall $t_1 = true$**:** $|if_s(true, t_2, t_3)|_P = 1 + |t_2|_P$, denn $if_s(true, t_2, t_3) \to_P t_2$, also $|if_s(true, t_2, t_3)|_P > |true|_P = 1$

Fall $t_1 = false$**:** analog zum vorherigen Fall

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_{bool}$**:** $\Rightarrow t$ ist nicht $\to_P$-minimal und $\mathsf{eval}_P(t) \neq \propto$ (s. o.) $\Rightarrow \exists q \in \mathcal{T}(\Sigma^c)_{bool}$ mit $t \to_P^+ q$ und q ist $\to_P$-minimal. Damit existiert dann aber auch die Auswertung $if_s(t_1, t_2, t_3) \to_P^+ if_s(true, t_2, t_3) \to_P t_2$, falls $q = true$, oder, falls $q = false$, die Auswertung $if_s(t_1, t_2, t_3) \to_P^+ if_s(false, t_2, t_3) \to_P t_3$, wobei $t_1 \to_P^+ q$ und $if_s(t_1, t_2, t_3) \to_P^+ if_s(true, t_2, t_3)$ bzw. $if_s(t_1, t_2, t_3) \to_P^+ if_s(false, t_2, t_3)$ die gleiche Länge besitzen. Damit erhalten wir $|if_s(t_1, t_2, t_3)|_P = |t_1|_P + 1 + |t_j|_P, j \in \{2, 3\}$, also $|if_s(t_1, t_2, t_3)|_P > |t_1|_P$.

Fall $f = eq_{nat}(t_1, t_2)$**:**

Fall $t_1, t_2 \in \mathcal{T}(\Sigma^c)_{nat}$**:** $\Rightarrow |eq_{nat}(t_1, t_2)|_P = 2$, denn $eq_{nat}(t_1, t_2) \to_P true$, falls $t_1 = t_2$ oder $eq_{nat}(t_1, t_2) \to_P false$ sonst, und $|t_1|_P = |t_2|_P = 1$.

Fall $i = 1$**:**

Fall $t_1 \in \mathcal{T}(\Sigma^c)_{nat}$**:** $\Rightarrow t_2 \notin \mathcal{T}(\Sigma^c)_{nat}$ (sonst obiger Fall) $\Rightarrow \exists t_2' \in \mathcal{T}(\Sigma^c)_{nat}.t_2 \to_P t_2'$ $\Rightarrow eq_{nat}(t_1, t_2) \to_P eq_{nat}(t_1, t_2') \Rightarrow |eq_{nat}(t_1, t_2)|_P > 1$ und $|t_1|_P = 1$.

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_{nat}$**:** $\Rightarrow t_1$ ist nicht $\to_P$-minimal und $\mathsf{eval}_P(t_1) \neq \propto$ (s. o.) $\Rightarrow \exists q \in \mathcal{T}(\Sigma^c)_{nat}$ mit $t \to_P^+ q$ und q ist $\to_P$-minimal. Damit existiert dann aber auch die Auswertung

$$eq_{nat}(t_1, t_2) \to_P^+ eq_{nat}(q, t_2) \to_P \begin{cases} s = true, & \text{falls } t_2 = q \in \mathcal{T}(\Sigma^c)_{nat} \\ s = false, & \text{falls } t_2 \neq q, t_2 \in \mathcal{T}(\Sigma^c)_{nat} \\ s = t_2', & \text{falls } t_2 \notin \mathcal{T}(\Sigma^c)_{nat} \text{ und } t_2 \to_P t_2' \end{cases}$$

Dabei besitzen $t_1 \to_P^+ q$ und $eq_{nat}(t_1, t_2) \to_P eq_{nat}(q, t_2)$ die gleiche Länge. So erhalten wir, da stets ein s mit $eq_{nat}(q, t_2) \to_P s$ existiert $|eq_{nat}(t_1, t_2)|_P > |t_1|_P$.

Fall $i = 2$**:** analog zu $i = 1$.

(b) Unter welcher Voraussetzung gilt die in (a) genannte Behauptung auch für den Fall $f = if_s$ und $i \neq 1$?

Lösungsvorschlag:

Sei $t = if_s(t_1, t_2, t_3)$. t_2 wird nur dann ausgewertet, falls t_1 zu TRUE ausgewertet wird, also falls $\mathsf{eval}_P(t_1) =$ TRUE. Gilt hingegen $\mathsf{eval}_P(t_1) =$ FALSE, so spielt die Länge der Auswertungsfolge zu t_2 keine Rolle für $|t|_p$. Die Auswertung von t kann somit „kurz“ sein, während die von t_2 „sehr lang“ sein kann.

Beispiel: $t = if_s(false, pred(pred(succ(succ(0)))), 0)$. $\Rightarrow |t|_p = 2$ ($t \to_P 0$), $|t_2|_p = 3$ ($t_2 \to_P pred(succ(0)) \to_p 0$).

Entsprechendes gilt für $i = 3$. Die Aussage aus Teilaufgabe (a) läßt sich somit erweitern und gilt auch für

(1.) $f = if_s, i = 2, \mathsf{eval}_P(t_1) = true$.
Beweis: $if_s(t_1, t_2, t_3) \to_P^* if_s(true, t_2, t_3) \to_P t_2 \Rightarrow |if_s(t_1, t_2, t_3)|_p > |t_2| \to_p t_2$.

(2.) $f = if_s, i = 3, eval_p(t_1) = false$. Beweis analog zum vorherigen Fall.

(c) Beweisen Sie: $|succ(t)|_P = |t|_P$

Lösungsvorschlag:

Für jeden Term $t \in \mathcal{T}(\Sigma^c)_{nat}$ mit $t \to_P t'$ gilt auch $succ(t) \to_P succ(t')$.

Fall $\mathsf{eval}_P(t) \neq \propto$**:** $\Rightarrow \exists q \in \mathcal{T}(\Sigma^c)_{nat}$ mit $t \to_P^* q$ und q ist $\to_P$-minimal.
Induktion über die Auswertungskosten $|t|_P$ von t:

Induktionsbasis $|t|_P = 1$**:** $\Rightarrow t \in \mathcal{T}(\Sigma^c)_{nat}$ (vgl. Satz 2.2.1.(i)) $\Rightarrow succ(t) \in \mathcal{T}(\Sigma^c)_{nat} \Rightarrow |succ(t)|_P = 1 = |t|_P$

Induktionsschritt: $t \notin \mathcal{T}(\Sigma^c)_{nat} \Rightarrow \exists r \in \mathcal{T}(\Sigma, \mathcal{V})_{nat}.t \to_P r \Rightarrow succ(t) \to_P succ(r) \Rightarrow |succ(t)|_P = 1 + |succ(r)|_P \underbrace{=}_{IH} 1 + |r|_P = |t|_P$.

Fall $\mathsf{eval}_P(t) = \propto$**:** $\Rightarrow$ es gibt eine unendliche Auswertungsfolge $\langle t_i \rangle_{i \in \mathbb{N}}$ mit $t_0 = t$ und $t_i \to_P t_{i+1}$ $\Rightarrow$ es existiert auch die unendliche Auswertungsfolge $\langle \hat{t}_i \rangle_{i \in \mathbb{N}}$ mit $\hat{t}_0 = succ(t)$ und $\hat{t}_i \to_P \hat{t}_{i+1}$, wobei $\hat{t}_i = succ(t_i)$ gilt. $\Rightarrow \mathsf{eval}_P(succ(t)) = \propto \Rightarrow |succ(t)|_P = |t|_P$.