Semantik und Programmverifikation

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

Übungsblatt mit Lösungsvorschlag 6

Aufgabe 6.1 (Datenstrukturen und funktionale Programmierung)

Gegeben sei die Datenstruktur $D_{sexpr} =$

$$\textbf{structure}\ \ atom(index : nat),\ \ nil,\ \ cons(car : sexpr, cdr : sexpr) : sexpr.$$

(a) Bestimmen Sie hierfür alle Regeln, die den allgemein angegebenen Regeln (1) bis (4) im Auswertungskalkül von Definition 2.5.4 entsprechen.

Lösungsvorschlag:

(1)

$$\frac{atom(n)}{atom(n')}\ (n \to_P n') \qquad \frac{cons(l,m)}{cons(l',m)}\ (l \to_P l') \qquad \frac{cons(l,m)}{cons(l,m')}\ (m \to_P m')$$

(2)

$$\frac{car(atom(n))}{nil}\ n \in \mathcal{T}(\Sigma^c(P))_{nat} \qquad \frac{cdr(atom(n))}{nil}\ n \in \mathcal{T}(\Sigma^c(P))_{nat}$$

$$\frac{index(cons(r,l))}{0}\ r,l \in \mathcal{T}(\Sigma^c(P))_{sexpr} \qquad \frac{index(nil)}{0} \qquad \frac{car(nil)}{nil} \qquad \frac{cdr(nil)}{nil}$$

(3)

$$\frac{index(atom(n))}{n}\ n \in \mathcal{T}(\Sigma^c(P))_{nat}$$

$$\frac{car(cons(l,m))}{l}\ l,m \in \mathcal{T}(\Sigma^c(P))_{sexpr} \qquad \frac{cdr(cons(l,m))}{m}\ l,m \in \mathcal{T}(\Sigma^c(P))_{sexpr}$$

(4)

$$\frac{index(a)}{index(a')}\ a \to_P a' \qquad \frac{car(l)}{car(l')}\ l \to_P l' \qquad \frac{cdr(l)}{cdr(l')}\ l \to_P l'$$

(b) Geben Sie eine Funktionsprozedur **function** $flatten(x : sexpr) : list \Leftarrow \ldots$ an, mit der jeder Binärbaum x in eine Liste überführt wird, die genau alle $index$–Bilder der Blätter von x enthält.

Lösungsvorschlag:

$F_{flatten} =$**function** $flatten\ (x : sexpr) : list\ \Leftarrow$
  if $x = nil$
    then $empty$
    else **if** $x = atom(index(x))$
      then $add(index(x), empty)$
      else $concat(flatten(car(x)), flatten(cdr(x))))$
    fi
  fi

$F_{concat} =$ **function** $concat\ (l, m : list) : list\ \Leftarrow$

if $l = empty$
then m
else $add(head(l), concat(tail(l), m))$
fi

Aufgabe 6.2 (Datenstrukturen und denotationale Semantik)

Erweitern Sie die Σ–Algebra D_P aus Definition 2.3.16 so, dass der Äquivalenzsatz 2.4.8 auch für funktionale Programme mit Datenstrukturen gilt. Geben Sie hierzu die modifizierte Definition für D_P an, ohne den Äquivalenzbeweis zu führen.

Lösungsvorschlag:

Sei $P = \langle D_1, \ldots, D_l, F_1, \ldots, F_k \rangle$ ein funktionales Programm mit Datenstrukturdefinitionen

$$D_i = \textbf{structure } \ldots, cons_{m_i}(sel_{m_i,1} : s_{m_i,1}, \ldots, sel_{m_i,n_{m_i}} : s_{m_i,n_{m_i}}), \ldots : s_i$$

und Funktionsprozeduren

$$F_j = \textbf{function } f_j(x^* : w_j) : s_j \Leftarrow R_{f_j}.$$

Zunächst erweitern wir D_{BM} um Deutungen für die Datenstrukturen: $D_{\mathsf{BM},P} = (D', \delta'_{BM})$ mit

$$D'_s = \mathcal{T}(\Sigma^c)_s \cup \{\varnothing_s\} \text{ für alle } s \in \mathcal{S}(P)$$

$$\delta'_{\mathsf{BM},f} := \delta_{\mathsf{BM},f} \text{ für alle } f \in \Sigma(\mathsf{BM})$$

$$\delta'_{\mathsf{BM},eq_s}(d,e) := \begin{cases} \varnothing_{bool} & , d = \varnothing_s \text{ oder } e = \varnothing_s \\ true & , d \neq \varnothing_s \wedge e \neq \varnothing_s \wedge d = e \\ true & , d \neq \varnothing_s \wedge e \neq \varnothing_s \wedge d \neq e \end{cases} \quad \text{für alle } s \in \mathcal{S}(P) \setminus \mathcal{S}(\mathsf{BM})$$

$$\delta'_{\mathsf{BM},if_s}(b,d,e) := \begin{cases} \varnothing_s & , b = \varnothing_{bool} \\ d & , b = true \\ e & , b = false \end{cases} \quad \text{für alle } s \in \mathcal{S}(P) \setminus \mathcal{S}(\mathsf{BM})$$

$$\delta'_{\mathsf{BM},cons_{m_i}}(d_1, \ldots, d_{n_{m_i}}) := \begin{cases} \varnothing_s & , d_h = \varnothing_{s_{m_i,h}} \text{ für ein } h \in \{1, \ldots, n_{m_i}\} \\ cons(d_1, \ldots, d_{n_{m_i}}) & , \text{ sonst} \end{cases}$$
$$\text{für alle } i \in \{1, \ldots, l\} \text{ und alle } m_i$$

$$\delta'_{\mathsf{BM},sel_{m_i,k}}(d) := \begin{cases} \varnothing_{s_{m_i,k}} & , d = \varnothing_{s_i} \\ \nabla_{s_{m_i,k}} & , d = cons_{m_i'}(q^*) \wedge m_i \neq m_i' \\ q_k & , d = cons_{m_i}(q_1, \ldots, q_{n_{m_i}}) \end{cases}$$
$$\text{für alle } i \in \{1, \ldots, l\}, \text{ alle } h \in \{1, \ldots, n_{m_i}\} \text{ und alle } m_i$$

Nun wird für jede Funktionsprozedur F_j ein Funktional

$$\mathfrak{R}_{F_j} : [D_{w_1} \to D_{s_1}] \times \cdots \times [D_{w_k} \to D_{s_k}] \to [D_{w_j} \to D_{s_j}]$$

mit $\mathfrak{R}_{F_j}[\![\phi]\!](d^*) := d'(\phi)[x_j^*/d^*](\mathfrak{R}_{F_j})$ definiert. Dabei ist $D'(\phi) = (D', \delta'(\phi))$ für $\phi = (\phi_1, \ldots, \phi_k)$ die $\Sigma(P)$-Expansion von $D_{\mathsf{BM},P}$ mit $\delta'(\phi)_{f_i} = \phi_i$ und $d'(\phi)$ eine $D'(\phi)$-Variablenbelegung. Analog zu den Definitionen 2.3.15 und 2.3.16 definieren wir:

$$\mathfrak{R}_P[\![\phi]\!] := (\mathfrak{R}_{P,F_1}[\![\phi]\!], \ldots, \mathfrak{R}_{P,F_k}[\![\phi]\!])$$

$$D_P := (D', \delta'_P) \text{ mit } \delta'_{P,f}(d^*) := \begin{cases} \mathfrak{R}_{F_j}[\![\mathit{fix}_{\mathfrak{R}_P}]\!](d^*) & , \text{ falls } f = f_j, j \in \{1,\ldots,k\} \\ \delta'_{\mathsf{BM},f}(d^*) & , \text{ sonst} \end{cases}$$

Aufgabe 6.3 (Parameterübergabeverfahren)

Gegeben sei das funktionale Programm $P = \langle F_{two} \rangle$ mit $F_{two} =$

function $two(x, y : nat) : nat \Leftarrow$ **if** $x = 0$ **then** 2 **else** $two(x - 1, two(x, y))$ **fi**.

Berechnen Sie die Ergebnisse der folgenden Ausdrücke durch Angabe der Auswertungsfolgen:

(a) $\mathsf{eval}_P(two(succ(0), succ(0)))$

Lösungsvorschlag:

$$\begin{array}{rl} & \underline{two(succ(0), succ(0))} \\ \Rightarrow_P & \underline{if(\underline{succ(0) = 0}, 2, two(pred(succ(0)), two(succ(0), succ(0))))} \\ \Rightarrow_P & \underline{if(false, 2, two(pred(succ(0)), two(succ(0), succ(0))))} \\ \Rightarrow_P & \underline{two(pred(succ(0)), two(succ(0), succ(0)))} \\ \Rightarrow_P & \underline{if(pred(succ(0)) = 0, 2, \ldots)} \\ \Rightarrow_P & \underline{if(0 = 0, 2, \ldots)} \\ \Rightarrow_P & \underline{if(true, 2, \ldots)} \\ \Rightarrow_P & 2 \\ \Rightarrow & \mathsf{eval}_P(two(succ(0), succ(0))) = 2 \end{array}$$

(b) $\mathsf{cbv\text{-}eval}_P(two(succ(0), succ(0)))$.

Lösungsvorschlag:

$$\begin{array}{rl} & \underline{two(succ(0), succ(0))} \\ \Rightarrow_P & \underline{if(\underline{succ(0) = 0}, 2, two(pred(succ(0)), two(succ(0), succ(0))))} \\ \Rightarrow_P & \underline{if(false, 2, two(pred(succ(0)), two(succ(0), succ(0))))} \\ \Rightarrow_P & two(\underline{pred(succ(0))}, \underline{two(succ(0), succ(0))}) \\ \Rightarrow_P & \ldots \\ \Rightarrow & \mathsf{cbv\text{-}eval}_P(two(succ(0), succ(0))) = \propto \end{array}$$