Semantik und Programmverifikation

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

Übungsblatt mit Lösungsvorschlag 8

Aufgabe 8.1 (Normal-rekursive und tail-rekursive Funktionen)

Bestimmen Sie alle normal-rekursiven und alle tail-rekursiven Funktionsprozeduren des funktionalen Programms $P_{hsort} \bigoplus S_{sort}$, wobei P_{hsort} und S_{sort} wie in Abschnitt 3.3 definiert ist.

Lösungsvorschlag:

tail-reskursiv: *ordered*, *member*, *perm*, *bottom*, *swap*, *min*, *key*, *completed*, *sort*

normal-reskursiv: alle tail-rekursiven sowie *pop*, $sift$, *insert*, *make_heap_structure*, *make_heap*, *heapsort*

Aufgabe 8.2 (Terminierungsanalyse)

Sei die Funktionsprozedur F_{funny} definiert durch

```
function funny(x : nat) : nat ⇐
    if x = 0
      then M
      else if funny(x − 1) = N
             then K
             else funny(x + 1)
           fi
    fi.
```

Beweisen oder widerlegen Sie für alle $M, N, K \in \{0, 1\}$ die Aussage: F_{funny} terminiert *stark*.

Hinweis: Verwenden Sie die Ergebnisse von Aufgabe 7.3, wenn möglich.

Lösungsvorschlag:

Fall $M = N = K = 0$**:** F_{funny} terminiert nicht stark.

Sei ϕ definiert durch $\phi(x) = 1$, sei $q = 1$. Dann gibt es *keine* fundierte Relation $\gg$ mit $1 = q \gg D(\phi)(q - 1) = 1$.

Fall $M = N = 0, K = 1$ **oder** $M = N = 1, K = 0$**:** F_{funny} terminiert nicht cbn (siehe Aufgabe 7.3) und damit auch nicht stark.

Fall $M = 0, N = 1$ **oder** $M = 1, N = 0$**:** F_{funny} terminiert nicht cbn (siehe Aufgabe 7.3) und damit auch nicht stark.

Fall $M = N = K = 1$**:** F_{funny} terminiert nicht stark. Beweis wie im ersten Fall, aber mit Fixpunkt $\phi(x) = 1$ und Gegenbeispiel $\phi(x) = 0$.

Aufgabe 8.3 (Semantik der Spezifikationssprache)

(a) Beweisen Sie, dass $[t \equiv M_P(t)] \in Th_P$ für alle $t \in \mathcal{T}(\Sigma(P))$ gilt.

Lösungsvorschlag:

Wir zeigen zunächst als Lemma $t \in \mathcal{T}(\Sigma^c) \Rightarrow M_P(t) = t$ durch strukturelle Induktion über t:

Basis $t \in \Sigma^c$**:** $\Rightarrow t = c$ für eine Konstante $c \Rightarrow M_P(t) = M_P(c) = \mu_c = \delta_{P,c} = c$

Schritt $t \notin \Sigma^c$**:** $t = f(t_1, \ldots, t_n) \Rightarrow M_P(t) = M_P(f(t_1, \ldots, t_n)) = \mu_f(M_P(t_1), \ldots, M_P(t_n)) = \delta_{P,f}(M_P(t_1), \ldots, M_P(t_n))$. Mit $t \in \mathcal{T}(\Sigma^c)$ gilt auch $t_i \in \mathcal{T}(\Sigma^c)$ mit $1 \leq i \leq n$, daher können wir nun die n IH $M_P(t_i) = t_i$ anwenden $\Rightarrow \ldots = \delta_{P,f}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) = t$

Sei nun $t \in \mathcal{T}(\Sigma(P))$. Dann ist $M_P(t) \in \mathcal{T}(\Sigma^c)$, denn M_P ist nur für terminierende P definiert. Mit obigem Lemma gilt nun $M_P(t) = M_P(M_P(t)) \Rightarrow M_P \models t \equiv M_P(t) \Rightarrow t \equiv M_P(t) \in Th(M_P) = Th_P$

(b) Beweisen Sie, dass eq_s und $\equiv$ für jedes terminierende Programm übereinstimmen, d. h., dass der folgende Satz gilt: „Sei P ein terminierendes funktionales Programm, $x^* \in \mathcal{V}_w$ und $t_1, t_2 \in \mathcal{T}(\Sigma(P), \mathcal{V}(x^*))_s$. Dann gilt:

$$[\forall x^* : w.\ t_1 \equiv t_2] \in Th_P \Leftrightarrow [\forall x^* : w.\ eq_s(t_1, t_2) \equiv true] \in Th_P\text{“}$$

Lösungsvorschlag:

	$[\forall x^* : w.\ t_1 \equiv t_2] \in Th_P$	Def. 3.4.1 (Th_P); Def 1.2.2 (Theorie)
$\Leftrightarrow$	$M_P \models \forall x^* : w.\ t_1 \equiv t_2$	Def. 1.2.2.(v)
$\Leftrightarrow$	$M_P[x^*/q^*](t_1) = M_P[x^*/q^*](t_2)$ für alle $q^* \in \mathcal{T}(\Sigma^c)_w$	da P termiert, gilt $M_P[x^*/q^*](t_1) \neq \varnothing \neq M_P[x^*/q^*](t_2)$; $M_P[x^*/q^*](t_1), M_P[x^*/q^*](t_2) \in \mathcal{T}(\Sigma^c)$ Def. 2.3.7, $\delta_{BM,eq}$
$\Leftrightarrow$	$\delta_{P,eq_s}(M_P[x^*/q^*](t_1), M_P[x^*/q^*](t_2)) = true$ für alle $q^* \in \mathcal{T}(\Sigma^c)_w$	Def. 3.4.1, μ_f
$\Leftrightarrow$	$\mu_{eq_s}(M_P[x^*/q^*](t_1), M_P[x^*/q^*](t_2)) = true$ für alle $q^* \in \mathcal{T}(\Sigma^c)_w$	Def. 1.2.1.(iii), 1.2.2
$\Leftrightarrow$	$M_P[x^*/q^*](eq_s(t_1, t_2)) = M_P[x^*/q^*](true)$ für alle $q^* \in \mathcal{T}(\Sigma^c)_w$	Def 1.2.2.(v)
$\Leftrightarrow$	$M_P \models \forall x^* : w.\ eq_s(t_1, t_2) \equiv true$	Def. 3.4.1 (Th_P); Def 1.2.2 (Theorie)
$\Leftrightarrow$	$[\forall x^* : w.\ eq_s(t_1, t_2) \equiv true] \in Th_P$	

(c) Beweisen Sie, dass eq_s und $\equiv$ für terminierende Programme nicht in $(AX_P)^{\models}$ übereinstimmen, d.h., dass der folgende Satz gilt: „Sei P ein terminierendes funktionales Programm. Dann existieren $x^* \in \mathcal{V}_w$ und $t_1, t_2 \in \mathcal{T}(\Sigma(P), \mathcal{V}(x^*))_s$ mit

$$[\forall x^* : w.\ t_1 \equiv t_2 \leftrightarrow \forall x^* : w.\ eq_s(t_1, t_2) \equiv true] \notin (AX_P)^{\models}\text{.“}$$

Lösungsvorschlag:

Sei $\phi := \forall x^* : w.\ t_1 \equiv t_2 \leftrightarrow \forall x^* : w.\ eq_s(t_1, t_2) \equiv true$. Dann gilt $\phi \notin (AX_P)^{\models}$, wenn es eine $\Sigma(P)$-Algebra A gibt, für die $A \models AX_P$ und $A \not\models \phi$ gilt. Wir wählen

$$
\begin{aligned}
P &:= \langle\rangle \\
\mathcal{A}_{bool} &:= \{\top, \bot\} \\
\mathcal{A}_{nat} &:= \mathbb{N} \cup \{\square\} \\
\alpha_{true} &:= \top \\
\alpha_{false} &:= \bot \\
\alpha_0 &:= 0 \\
\alpha_{succ}(n) &:= \begin{cases} \square & \text{falls } n = \square \\ n+1 & \text{sonst} \end{cases} \\
\alpha_{pred}(n) &:= \begin{cases} \square & \text{falls } n = \square \\ 0 & \text{falls } n = 0 \\ n-1 & \text{sonst} \end{cases} \\
\alpha_{eq_{nat}}(n_1, n_2) &:= \begin{cases} \bot & \text{falls } n_1 = \square \text{ oder } n_2 = \square \text{ oder } n_1 \neq n_2 \\ \top & \text{sonst} \end{cases} \\
\alpha_{if_{nat}}(b, n_1, n_2) &:= \begin{cases} n_1 & \text{falls } b = \top \\ n_2 & \text{falls } b = \bot \end{cases}
\end{aligned}
$$

Dann gilt $A \models AX_P$ bzw. $A \models AX_{BM}$.

Für ϕ wählen wir nun $t_1 := x$, $t_2 := x$ und erhalten

- $A \models [\forall x : nat.\ x \equiv x]$, denn (vgl. Def. 1.2.2.(v)) $A[x/n](x) = A[x/n](x)$ für alle $n \in \mathcal{A}_{nat}$, $\Leftrightarrow n = n$ für alle $n \in \mathcal{A}_{nat}$.
- $A \not\models [\forall x : nat.\ eq_{nat}(x, x) \equiv true]$, denn $\exists n \in \mathcal{A}_{nat}$ mit $A[x/n](eq_{nat}(x, x)) \neq A[x/n](true)$: Wir wählen $n = \square$, so gilt $\alpha_{eq_{nat}}(\square, \square) = \bot$. Weiter gilt $\alpha_{true} = \top$. Da aber $\top \neq \bot$, gilt $\alpha_{eq_{nat}}(\square, \square) \neq \alpha_{true}$.

Insgesamt ergibt sich also $A \not\models \phi$ bzw. $\phi \notin (AX_P)^{\models}$.

Anmerkung 1: Es gilt $A \models \neg\bot \equiv \top$, also $\neg\bot \equiv \top \in Th(A)$. Wäre nämlich $\neg\bot \equiv \top \notin Th(A)$, dann wäre $\bot \equiv \top \in Th(A)$ (die Theorie ist vollständig). Aus α_{if} könnte man dann aber leicht einen Widerspruch konstruieren (z. B. mit $if_{nat}(true, 0, pred(0))$. Da $Th(A)$ konsistent ist, verbietet sich das jedoch.

Anmerkung 2: Wir nutzen hier aus, dass A nicht termerzeugt ist, es also ein zusätzliches Trägerelement $\square$ gibt. A ist damit nicht isomorph zu M_P. Für dieses Trägerelement können wir $\alpha_{eq_{nat}}$ beliebig wählen, ohne AX_{BM} zu verletzen.

(vgl. Übung 3.4.1).

Aufgabe 8.4 (Korrektheitsbeweis durch Induktion)

Sei $P_{\text{double}} = \langle F_{\text{ge}}, F_{\text{half}}, F_{\text{double}} \rangle$ das funktionale Programm mit

$$\begin{aligned}
F_{\text{ge}} = \ & \textbf{function } ge(x, y : nat) : bool \Leftarrow \\
& \quad \textbf{if } y = 0 \textbf{ then } true \textbf{ else } (\textbf{if } x = 0 \textbf{ then } false \textbf{ else } ge(x-1, y-1) \textbf{ fi}) \textbf{ fi}, \\
F_{\text{half}} = \ & \textbf{function } half(x : nat) : nat \Leftarrow \\
& \quad \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } (\textbf{if } x = 1 \textbf{ then } 0 \textbf{ else } 1 + half(x-2) \textbf{ fi}) \textbf{ fi}, \\
F_{\text{double}} = \ & \textbf{function } double(x : nat) : nat \Leftarrow \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } 2 + double(x-1) \textbf{ fi}.
\end{aligned}$$

Definieren Sie eine unendliche entscheidbare Menge $\mathsf{IND}_{P_{\text{double}}}$ von Induktionsaxiomen für P_{double}. Zeigen Sie $\mathsf{IND}_{P_{\text{double}}} \subseteq Th_{P_{\text{double}}}$ und beweisen Sie mit Hilfe von $\mathsf{IND}_{P_{\text{double}}}$

$$[\forall x : nat.\ ge(x, double(half(x))) \equiv true] \in Th_{P_{double}}.$$

(vgl. Übung 3.7.2)

Lösungsvorschlag:

Wir wählen

$$\begin{aligned}
\mathsf{IND}_P := \big\{ & \psi[0] \wedge \\
& \psi[1] \wedge \\
& \forall x : nat.\ \big(\psi[x] \to \psi[succ^2(x)]\big) \to \forall x : nat. \psi[x] \\
& \mid \psi[x] \in \mathcal{F}(\Sigma, \mathcal{V}) \wedge \mathcal{V}_f(\psi[x]) = \{x\} \ \big\}.
\end{aligned}$$

Zeige $\mathsf{IND}_P \in Th_{P_{double}}$**:** Die Relation $<_2 \subseteq \mathcal{T}(\Sigma^c) \times \mathcal{T}(\Sigma^c)$ mit $q <_2 succ^2(q)$ ist fundiert also gilt das Noethersche Induktionsschema. Damit folgt

$$\begin{aligned}
& \Big[\forall x \in \mathcal{T}(\Sigma^C). \big(\forall y \in \mathcal{T}(\Sigma^C). (y <_2 x \Rightarrow \psi[y] \in Th_{P_{double}}) \Rightarrow \psi[x] \in Th_{P_{double}}\big)\Big] \\
& \Rightarrow \forall x \in \mathcal{T}(\Sigma^C). \psi[x] \in Th_{P_{double}}
\end{aligned}$$

Die Minimalen Elemente bezüglich $<_2$ sind 0 und 1, also folgt

$$\begin{aligned}
& \Big[\psi[0] \in Th_{P_{double}} \wedge \psi[1] \in Th_{P_{double}} \wedge \big(\forall x \in \mathcal{T}(\Sigma^C). \psi[x] \in Th_{P_{double}} \Rightarrow \psi[succ^2(x)] \in Th_{P_{double}}\big)\Big] \\
& \Rightarrow \forall x \in \mathcal{T}(\Sigma^C). \psi[x] \in Th_{P_{double}}
\end{aligned}$$

Somit gilt

$$\left[\Big(\psi[0] \wedge \psi[1] \wedge \big(\forall x \in \mathcal{T}(\Sigma^C). \psi[x] \to \psi[succ^2(x)]\big)\Big) \to \forall x \in \mathcal{T}(\Sigma^C). \psi[x]\right] \in Th_{P_{double}}$$

Also gilt $\mathsf{IND}_P \in Th_{P_{double}}$.

Zeige $\big(\forall x : nat. ge(x, double(half(x))) \equiv true\big) \in Th_{P_{double}}$ **:**

1) $ge(0, double(half(0))) \equiv true \in Th_{P_{double}}$

$$\delta_{ge}(0, \delta_{double}(\delta_{half}(0))) = \delta_{ge}(0, \delta_{double}(0)) = \delta_{ge}(0, 0) = true$$

2) $ge(1, double(half(1))) \equiv true \in Th_{P_{double}}$

$$\delta_{ge}(1, \delta_{double}(\delta_{half}(1))) = \delta_{ge}(1, \delta_{double}(0)) = \delta_{ge}(1, 0) = true$$

3) $\big(\forall x : nat\ ge(x, double(half(x))) \equiv true \rightarrow ge(succ^2(x), double(half(x))) \equiv true\big) \in Th_{P_{double}}$

$\Leftrightarrow \big[\forall x \in \mathcal{T}(\Sigma^C)_{nat}.\, \delta_{ge}(x, \delta_{double}(\delta_{half}(x))) = true \Rightarrow \delta_{ge}(\delta^2_{succ}(x), \delta_{double}(\delta_{half}(\delta^2_{succ}(x)))) = true\big]$

Sei also $x \in \mathcal{T}(\Sigma^C)_{nat}$ beliebig mit $\delta_{ge}(x, \delta_{double}(\delta_{half}(x))) = true$, dann gilt

$$\begin{aligned}
&\delta_{ge}(\delta^2_{succ}(x), \delta_{double}(\delta_{half}(\delta^2_{succ}(x))))\\
&= \delta_{ge}(\delta^2_{succ}(x), \delta_{double}(\delta_{succ}(\delta_{half}(x))))\\
&= \delta_{ge}(\delta^2_{succ}(x), \delta^2_{succ}(\delta_{double}(\delta_{half}(x))))\\
&= \delta_{ge}(x, \delta_{double}(\delta_{half}(x)))\\
&= true
\end{aligned}$$

Aufgabe 8.5 (Nicht-Äquivalenzmodell)

Definieren Sie ein Nicht-Äquivalenzmodell N von AX_{BM} mit $N \not\models \forall x : nat.\ \neg x \equiv succ(x)$ (vgl. Übung 3.6.1.(i)).

Lösungsvorschlag:

Sei $N = (\mathcal{N}, \nu)$ mit

$$\begin{aligned}
&\mathcal{N}_{bool} = \{\top, F\}\\
&\mathcal{N}_{nat} = \mathbb{N} \cup \{\square\},\\
&\nu_{true} = T,\ \nu_{false} = F,\ \nu_0 = 0,\\
&\nu_{succ}(n) = \begin{cases} n+1, & n \in \mathbb{N} \\ \square, & \text{sonst} \end{cases},\\
&\nu_{pred}(n) = \begin{cases} n-1, & n \in \mathbb{N} \setminus \{0\} \\ 0, & n = 0 \\ \square, & \text{sonst} \end{cases},\\
&\nu_{eq}(n, m) = \begin{cases} T, & n = m \\ F, & \text{sonst} \end{cases},\\
&\nu_{if}(b, n, m) = \begin{cases} n, & b = T \\ m, & b = F \end{cases}
\end{aligned}$$

Offensichtlich gilt $N \models AX_{\mathsf{BM}}$. Weiter gilt $\square = \delta_{succ}(\square) \Rightarrow N \not\models \forall x : nat.\ \neg x \equiv succ(x) \Rightarrow N \not\models Th_P \Rightarrow$ N ist Nicht-Äquivalenzmodell.

Aufgabe 8.6 (Programme, Axiome, Modelle)

Sei das funktionale Programm $P = \langle F_{plus} \rangle$ gegeben durch

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

Sei weiter die Σ-Algebra $A = (\mathcal{A}, \alpha)$ gegeben durch

$$
\begin{array}{rcl}
\mathcal{A}_{bool} & := & \{\top, \bot\} \\
\mathcal{A}_{nat} & := & \{2^i \mid i \in \mathbb{N}\} \\
\alpha_{true} & := & \top \\
\alpha_{false} & := & \bot \\
\alpha_0 & := & 1 \\
\alpha_{succ}(n) & := & n * 2 \\
\alpha_{pred}(n) & := & \begin{cases} n/2, \text{ falls } n \neq 1 \\ 1, \text{ falls } n = 1 \end{cases} \\
\alpha_{eq}(n,m) & := & \begin{cases} \top, \text{ falls } n = m \\ \bot, \text{ falls } n \neq m \end{cases} \\
\alpha_{if}(b,n,m) & := & \begin{cases} n, \text{ falls } b = \top \\ m, \text{ falls } b = \bot \end{cases} \\
\alpha_{plus}(n,m) & := & n * m
\end{array}
$$

Beweisen oder widerlegen Sie:

(a) A ist ein Äquivalenzmodell von $AX_{P_{plus}}$.

(b) A ist ein Standardmodell von $AX_{P_{plus}}$.

Lösungsvorschlag:

Wir zeigen, $M_P \simeq_\Sigma A$. Dazu definieren wir $\phi : \mathcal{M} \to \mathcal{A}$ mit

$$\phi(true) = \top, \phi(false) = \bot \text{ und } \phi(succ^n(0) = 2^n.$$

Die Abbildung ist offensichtlich bijektiv. Wir zeigen, dass ϕ ein Σ-Homomorphismus ist:

$$\phi(\mu_{false}) = \phi(false) = \bot = \alpha_{false}$$

$$\phi(\mu_{true}) = \phi(true) = \top = \alpha_{true}$$

$$\phi(\mu_0) = \phi(0) = 1 = \alpha_0$$

$$\phi(\mu_{succ}(succ^n(0))) = \phi(succ^{n+1}(0)) = 2^{n+1} = \alpha_{succ}(2^n) = \alpha_{succ}(\phi(succ^n(0)))$$

$$\phi(\mu_{pred}(0)) = \phi(0) = 1 = \alpha_{pred}(1) = \alpha_{pred}(\phi(0))$$

$$\phi(\mu_{pred}(succ^{n+1}(0))) = \phi(succ^n(0)) = 2^n = \alpha_{pred}(2^{n+1}) = \alpha_{pred}(\phi(succ^{n+1}(0)))$$

$$\phi(\mu_{plus}(succ^n(0), succ^m(0))) = \phi(succ^{n+m}(0)) = 2^{n+m} = \alpha_{plus}(2^n, 2^m) = \alpha_{plus}(\phi(succ^n(0)), \phi(succ^m(0)))$$

$$\phi(\mu_{eq}(succ^n(0), succ^m(0))) = \phi(false) = \bot = \alpha_{eq}(2^n, 2^m) = \alpha_{eq}(\phi(succ^n(0)), \phi(succ^m(0))) \text{ wenn } m \neq n$$

$$\phi(\mu_{eq}(succ^n(0), succ^n(0))) = \phi(true) = \top = \alpha_{eq}(2^n, 2^n) = \alpha_{eq}(\phi(succ^n(0)), \phi(succ^n(0)))$$

$$\phi(\mu_{if}(\mu_{false}, succ^m(0), succ^n(0))) = \phi(succ^n(0)) = 2^n = \alpha_{if}(\bot, 2^m, 2^n) = \alpha_{if}(\phi(\mu_{false}), \phi(succ^m(0)), \phi(succ^n(0)))$$

$$\phi(\mu_{if}(\mu_{true}, succ^m(0), succ^n(0))) = \phi(succ^m(0)) = 2^m = \alpha_{if}(\top, 2^m, 2^n) = \alpha_{if}(\phi(\mu_{true}), \phi(succ^m(0)), \phi(succ^n(0)))$$

Damit ist ϕ ein Σ-Isomorphismus und es gilt $M_P \simeq_\Sigma A$. Also ist A ein Äquivalenzmodell und ein Standardmodell.