Semantik und Programmverifikation

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

Übungsblatt mit Lösungsvorschlag 3

Aufgabe 3.1 (Rekursionsrelation)

Beweisen Sie die Terminierung des folgenden Algorithmus und geben Sie an, welche Funktion dieser berechnet.

```
function g (x, y, z : ℕ) : ℕ ⇐
  if z = 0
    then if x = 0
           then 0
           else g(x − 1, y, y)
         fi
    else g(x, y, z − 1) + 1
  fi
end
```

Hinweis: Zeigen Sie die Terminierung, indem Sie eine Rekursionsrelation $>_g$ definieren und deren Fundiertheit nachweisen (vgl. Beispiele 1.4.1 und 1.4.2 im Buch).

Lösungsvorschlag:

Wir definieren zunächst die Relation $R \subseteq \mathbb{N}^3$ und zeigen ihre Fundiertheit wie folgt:

(a) $(\mathbb{N}, >_{\mathbb{N}})$ ist nach Beispiel 1.3.1.(i) fundiert.

(b) Damit ist auch $Q \subseteq \mathbb{N}^2$ mit

$$((m_1, n_1), (m_2, n_2)) \in Q :\Leftrightarrow m_1 >_{\mathbb{N}} m_2 \text{ oder } (m_1 = m_2 \text{ und } n_1 >_{\mathbb{N}} n_2)$$

nach Satz 1.4.1.(iii) fundiert.

(c) Wir definieren die Abbildung $f : \mathbb{N}^3 \to \mathbb{N}^2$ mit $f : (x, y, z) \mapsto (x, z)$ und die Relation $R \subseteq \mathbb{N}^3$ wie folgt:

$$((x_1, y_1, z_1), (x_2, y_2, z_2)) \in R :\Leftrightarrow (f(x_1, y_1, z_1), f(x_2, y_2, z_2)) \in Q$$

Nach Satz 1.4.1.(ii) ist dann auch R fundiert. Es gilt:

$$((x_1, y_1, z_1), (x_2, y_2, z_2)) \in R \Leftrightarrow x_1 >_{\mathbb{N}} x_2 \text{ oder } (x_1 = x_2 \text{ und } z_1 >_{\mathbb{N}} z_2)$$

Die Betrachtung der rekursiven Aufrufe in g ergibt für $>_g$

1. $(x + 1, y, 0) \quad >_g \quad (x, y, y)$ für alle $x, y \in \mathbb{N}$
2. $(x, y, z + 1) \quad >_g \quad (x, y, z)$ für alle $x, y, z \in \mathbb{N}$

Wegen $x + 1 >_{\mathbb{N}} x$ (in 1.) und $z + 1 >_{\mathbb{N}} z$ (in 2.) ist $>_g \subseteq R$. Mit Satz 1.4.1.(i) folgt dann aus der Fundiertheit von R, dass $>_g$ fundiert ist. Also terminiert g für alle Eingaben $(x, y, z) \in \mathbb{N}^3$.

Der Algorithmus berechnet übrigens $g : (x, y, z) \mapsto x \cdot y + z$.

Aufgabe 3.2 (funktionales Programmieren)

Schreiben Sie ein funktionales Programm $P \in \mathcal{FP}$, das

- eine Funktionsprozedur function $exp(x, y : nat) : nat \Leftarrow \ldots$ zur Berechnung der Exponentialfunktion und
- eine Funktionsprozedur function $prim(x : nat) : bool \Leftarrow \ldots$ zur Überprüfung, ob eine natürliche Zahl prim ist,

enthält.

Lösungsvorschlag:

$P = \langle F_{\text{zero}}, F_{\text{ge}}, F_{\text{plus}}, F_{\text{times}}, F_{\text{exp}}, F_{\text{minus}}, F_{\text{teiler}}, F_{\text{prim2}}, F_{\text{prim}} \rangle$.

$F_{\text{zero}}, F_{\text{ge}}, F_{\text{plus}}, F_{\text{times}}$ sind wie in Beispiel 2.1.1 definiert.

$F_{\text{exp}} =$

```
function exp(x, y: nat): nat <=
  if zero(y)
     then 1
     else times(x, exp(x, pred(y)), x)
  fi
```

$F_{\text{minus}} =$

```
function minus(x, y: nat): nat <=
  if zero(y)
     then x
     else if zero(x)
             then 0
             else minus(pred(x), pred(y))
          fi
  fi
```

$F_{\text{teiler}} =$

```
function teiler(x, y: nat): bool <=
  if zero(x)
     then false
     else if x=y
             then true
             else if ge(x, y)
                     then false
                     else teiler(x, minus(y, x))
                  fi
          fi
  fi
```

$F_{\text{prim2}} =$

```
function prim2(x, y: nat): bool <=
  if ge(1, y)
     then true
     else if teiler(y, x)
```

```
            then false
            else prim2(x, pred(y))
         fi
  fi
```

$F_{\text{prim}} =$

```
function prim(x: nat): bool <=
  if zero(x)
     then false
     else if zero(pred(x))
            then false
            else prim2(x, pred(x))
          fi
  fi
```

F_{exp} berechnet x^y, F_{minus} berechnet $x - y$, F_{teiler} ermittelt, ob x Teiler von y ist, F_{prim} untersucht, ob x eine Primzahl ist, F_{prim2} ist eine Hilfsfunktion für F_{prim} und testet, ob alle Zahlen zwischen 2 und $x - 1$ keine Teiler von x sind oder ob eine dieser Zahlen x teilt.

Aufgabe 3.3 (Auswertungsrelation der Basismaschine)

(a) Vervollständigen Sie den Beweis zu Satz 2.2.1 (i), in dem Sie im Induktionsschritt die Fälle $t = pred(t_1)$ und $t = eq_{nat}(t_1, t_2)$ betrachten.

Lösungsvorschlag:

Fall $t = pred(t_1)$**:** IH lautet: $t_1 \in \mathcal{T}(\Sigma^c) \Leftrightarrow t_1 \in \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$.

$t \notin \mathcal{T}(\Sigma^c)$, also zu zeigen: $pred(t_1) \notin \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$

Fall $t_1 = 0$**:** $\Rightarrow t \rightarrow_{BM} 0 \Rightarrow t \in \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$

Fall $t_1 = succ^n(0)$ **mit** $n > 0$**:** $\Rightarrow t \rightarrow_{BM} succ^{n-1}(0)$

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_{nat}$**:** $\Rightarrow$ (mit IH) $\exists r_1 \in \mathcal{T}(\Sigma(BM))_{nat}$ mit $t_1 \rightarrow_{BM} r_1 \Rightarrow t \rightarrow_{BM} pred(r_1)$.

Fall $t = eq(t_1, t_2)$**:** IH ist $\forall i \in \{1, 2\}.t_i \in \mathcal{T}(\Sigma^c) \Leftrightarrow t_i \in \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$

$t \notin \mathcal{T}(\Sigma^c)$, also zu zeigen: $t \notin \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$.

Fall $t_1, t_2 \in \mathcal{T}(\Sigma^c)$**:** $\Rightarrow t \rightarrow_{BM} b$ mit $b \in \{\text{TRUE}, \text{FALSE}\} \Rightarrow t \notin \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$

Fall $t_1 \notin \mathcal{T}(\Sigma^c)$**:** mit IH $\Rightarrow \exists r_1 \in \mathcal{T}(\Sigma(BM)))$ mit $t_1 \rightarrow_{BM} r_1 \Rightarrow t \rightarrow_{BM} eq(r1, t_2) \Rightarrow t \notin \min_{\rightarrow_{BM}}(\mathcal{T}(\Sigma(BM)))$.

Fall $t_2 \notin \mathcal{T}(\Sigma^c)$**:** analog zum vorherigen Fall.

(b) Vervollständigen Sie den Beweis zu Satz 2.2.1 (ii), in dem Sie im Induktionsschritt die Fälle $t = pred(t_1)$ und $t = if_s(b, p_1, p_2)$ betrachten.

Lösungsvorschlag:

Fall $t = pred(t_1)$**:** Annahme: es gibt s_1, s_2 mit $t \rightarrow_{BM} s_1$ und $t \rightarrow_{BM} s_2$.

Fall $t_1 = 0$**:** $\Rightarrow s_1 = s_2 = 0$, da nur Regel 2 anwendbar ist.

Fall $t_1 = succ^n(0)$**:** Da $t_1 \in \mathcal{T}(\Sigma^c)$ und damit $\rightarrow_{BM}$-minimal ist, kann Regel 4 nicht angewendet werden. Anwendbar ist also nur Regel 3, für alle s_1, s_2 mit $t \rightarrow_{BM} s_1$ und $t \rightarrow_{BM} s_2$ gilt also $s_1 = s_2$

Fall $t_1 \notin \mathcal{T}(\Sigma^c)_{nat}$**:** Damit ist t_1 nicht $\rightarrow_{BM}$-minimal, es gibt also ein r' mit $t_1 \rightarrow_{BM} r'$. Auf t ist also nur Regel 4 anwendbar, für alle s_1, s_2 mit $t \rightarrow_{BM} s_1$ und $t \rightarrow_{BM} s_2$ gilt also $s_1 = s_2$

Fall $t = if(b, p_1, p_2)$**:**

Fall $b \in \{\textsf{true}, \textsf{false}\}$**:** $\Rightarrow s_1 = p_i, s_2 = p_2$ mit $i \in \{1, 2\}$ (Regel 9 und 10 sind die einzig anwendbaren) $\Rightarrow s_1 = s_2$

Fall $b \notin \{\textsf{true}, \textsf{false}\}$**:** $\Rightarrow s_1 = if(b_1, p_1, p_2), s_2 = if(b_2, p_1, ps_2)$ (Regel 11 einzig anwendbare Regel) $\Rightarrow b \rightarrow_{BM} b_1$ und $b \rightarrow_{BM} b_2$

es gilt $t >_{\mathcal{T}} b$, daher ist für b die IH anwendbar. $\Rightarrow b_1 \rightarrow_{BM} r$ und $b_2 \rightarrow_{BM} r$ oder $b_1 = b_2$ $\Rightarrow if(b_1, p_1, p_2) \rightarrow_{BM} if(r, p_1, p_2)$ und $if(b_2, p_1, p_2) \rightarrow_{BM} if(r, p_1, p_2)$ oder $if(b_1, p_1, p_2) = if(b_2, p_1, p_2)$.

Aufgabe 3.4 (Auswertungsrelation für Programme)

Beweisen Sie die folgende Behauptung:

Für jede Regel der Form

$$\frac{f(t_1, \ldots, t_j, \ldots, t_n)}{f(t_1, \ldots, r_j, \ldots, t_n)}, \text{ falls } t_j \longrightarrow_P r_j$$

im Auswertungskalkül für funktionale Programme gilt

$$f(t_1, \ldots, t, \ldots, t_n) \longrightarrow_P^* f(t_1, \ldots, r, \ldots, t_n), \text{ falls } t \longrightarrow_P^* r.$$

Lösungsvorschlag:

zu zeigen: $\forall t, r \in \mathcal{T}(\Sigma, \mathcal{V}). \forall t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V}). t \rightarrow_P^* r \Rightarrow f(t_1, \ldots, t, \ldots, t_n) \rightarrow_P^* f(t_1, \ldots, r, \ldots, t_n)$

Wir beweisen durch Induktion über n (Länge der Ableitung) die äquivalente Aussage:

$$\forall n \in \mathbb{N}. \underbrace{\forall t, r \in \mathcal{T}(\Sigma, \mathcal{V}). \forall t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V}). t \rightarrow_P^n r \Rightarrow f(t_1, \ldots, t, \ldots, t_n) \rightarrow_P^* f(t_1, \ldots, r, \ldots, t_n)}_{P(n)}$$

Basisfall $P(0)$**:** Wir nehmen also an $t \rightarrow_P^0 r$. $\Rightarrow t = r \Rightarrow f(t_1, \ldots, t, \ldots, t_n) = f(t_1, \ldots, r, \ldots, t_n) \Rightarrow$ $f(t_1, \ldots, t, \ldots, t_n) \rightarrow_P^* f(t_1, \ldots, r, \ldots, t_n)$

Schrittfall $P(n) \Rightarrow P(n+1)$**:** Annahme: $t \rightarrow_P^{n+1} r \Rightarrow \exists q \in \mathcal{T}(\Sigma, \mathcal{V})$ mit $t \rightarrow_P^n q$ und $q \rightarrow_P^1 r$

$$\begin{array}{llll} \Rightarrow f(t_1, \ldots, t, \ldots, t_n) & \rightarrow_P^* & f(t_1, \ldots, q, \ldots, t_n) & (1) \text{ mit IH} \\ \Rightarrow f(t_1, \ldots, q, \ldots, t_n) & \rightarrow_P^1 & f(t_1, \ldots, r, \ldots, t_n) & (2) \text{ mit Kalkülregel} \end{array}$$

mit (1) und (2) folgt $f(t_1, \ldots, t, \ldots, t_n) \rightarrow_P^* f(t_1, \ldots, r, \ldots, t_n)$.

Aufgabe 3.5 (Auswertungsfolgen)

Das Programm P bestehe aus den folgenden Funktionsprozeduren:

function $zero(x : nat) : bool \Leftarrow eq(x, 0)$

function $plus(x, y : nat) : nat \Leftarrow$
$if_{nat}(zero(x), y, succ(plus(pred(x), y)))$

function $times(x, y : nat) : nat \Leftarrow$
$if_{nat}(zero(x), 0, plus(times(pred(x), y), y))$

function $inner(x : nat) : nat \Leftarrow inner(succ(x))$

function $outer(x : nat) : nat \Leftarrow succ(outer(x))$

Berechnen Sie durch Angabe der Auswertungsfolgen die Ergebnisse von

(a) $\mathsf{eval}_P(times(0, outer(0)))$,

Lösungsvorschlag:

	$\underline{times(0, outer(0))}$	12
$\rightarrow_P$	$if(\underline{zero(0)}, 0, plus(times(pred(0), outer(0)), outer(0)))$	12
$\rightarrow_P$	$if(\underline{eq(0,0)}, 0, plus(times(pred(0), outer(0)), outer(0)))$	5
$\rightarrow_P$	$\underline{if(true, 0, plus(times(pred(0), outer(0)), outer(0)))}$	9
$\rightarrow_P$	0	
$\Rightarrow$	$\mathsf{eval}_P(times(0, outer(0))) = 0$	

(b) $\mathsf{eval}_P(times(outer(0), 0))$,

Lösungsvorschlag:

	$\underline{times(outer(0), 0)}$	12
$\rightarrow_P$	$if(\underline{zero(outer(0))}, 0, plus(times(pred(outer(0)), 0), 0))$	12
$\rightarrow_P$	$if(eq(\underline{outer(0)}, 0), 0, plus(times(pred(outer(0)), 0), 0))$	12
$\rightarrow_P$	$if(eq(\underline{succ(outer(0))}, 0), 0, plus(times(pred(outer(0)), 0), 0))$	12
$\rightarrow_P$	$if(eq(succ(\underline{succ(outer(0))}), 0), 0, plus(times(pred(outer(0)), 0), 0))$	12
$\rightarrow_P$	$\ldots$	
$\Rightarrow$	$\mathsf{eval}_P(times(outer(0), 0)) = \propto$	

(c) $\mathsf{eval}_P(times(succ(0), outer(0)))$,

Lösungsvorschlag:

$$
\begin{array}{lll}
 & \underline{times(succ(0), outer(0))} & 12 \\
\to_P & if(\underline{zero(succ(0))}, 0, plus(times(pred(succ(0)), outer(0)), outer(0))) & 12 \\
\to_P & if(\underline{eq(succ(0), 0)}, 0, plus(times(pred(succ(0)), outer(0)), outer(0))) & 6 \\
\to_P & \underline{if(false, 0, plus(times(pred(succ(0)), outer(0)), outer(0)))} & 10 \\
\to_P & \underline{plus(times(pred(succ(0)), outer(0)), outer(0))} & 12 \\
\to_P & if(\underline{zero(times(pred(succ(0)), outer(0)))}, outer(0), & \\
 & succ(plus(pred(times(pred(succ(0)), outer(0))), outer(0)))) & 12 \\
\to_P & if(eq(\underline{times(pred(succ(0)), outer(0))}, 0), outer(0), & \\
 & succ(plus(pred(times(pred(succ(0)), outer(0))), outer(0)))) & 12 \\
\to_P & if(eq(if(\underline{zero(pred(succ(0)))}, 0, plus(times(pred(pred(succ(0))), outer(0)), outer(0))), 0), & \\
 & outer(0), succ(plus(pred(times(pred(succ(0)), outer(0))), outer(0)))) & \\
\to_P & \ldots & ausgelassen \\
\to_P & \underline{outer(0)} & 12 \\
\to_P & \underline{succ(outer(0))} & 12 \\
\to_P & succ(\underline{succ(outer(0))}) & 12 \\
\to_P & \ldots & \\
\Rightarrow & \mathsf{eval}_P(times(succ(0), outer(0))) = \propto &
\end{array}
$$

(d) $\mathsf{eval}_P(eq_{nat}(inner(0), outer(0)))$,

Lösungsvorschlag:

$$
\begin{array}{lll}
 & eq(\underline{inner(0)}, outer(0)) & 12 \\
\to_P & eq(\underline{inner(succ(0))}, outer(0)) & 12 \\
\to_P & eq(\underline{inner(succ(succ(0)))}, outer(0)) & 12 \\
\to_P & \ldots & \\
\Rightarrow & \mathsf{eval}_P(eq(inner(0), outer(0))) = \propto &
\end{array}
$$

(e) $\mathsf{eval}_P(if_{nat}(eq_{nat}(inner(0), inner(0)), 0, 0))$.

Lösungsvorschlag:

$$
\begin{array}{lll}
 & if(eq(\underline{inner(0)}, inner(0)), 0, 0) & 12 \\
\to_P & if(eq(\underline{inner(succ(0))}, inner(0)), 0, 0) & 12 \\
\to_P & if(eq(\underline{inner(succ(succ(0)))}, inner(0)), 0, 0) & 12 \\
\to_P & \ldots & \\
\Rightarrow & \mathsf{eval}_P(if(eq(inner(0), inner(0)), 0, 0)) = \propto &
\end{array}
$$