Technische Universität München **WS 2006/07**
Institut für Informatik
Prof. Manfred Broy
A. Gruler, B. Spanfelner **Musterlösung zu Blatt 7**

Übungen zu „Formale Methoden“

Aufgabe 1 Rekursive Funktionen [LÖSUNG]

Wir beweisen, dass die Funktion $\mathsf{sum}(t, s)$ für alle Parameterwerte stets terminiert.

$$
\begin{array}{l}
\underline{\textbf{fct}}\ \mathsf{sum} = (t\text{: Tree Nat},\ s\text{: Seq Tree Nat})\ \mathsf{Nat} \\
\quad \underline{\textbf{if}}\ t = \mathsf{etree}\quad \underline{\textbf{then}}\ \underline{\textbf{if}}\ s = \langle\rangle\ \underline{\textbf{then}}\ 0 \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \underline{\textbf{else}}\ \mathsf{sum}(\mathsf{first}(s), \mathsf{rest}(s)) \\
\qquad\qquad\qquad\qquad\qquad \underline{\textbf{fi}} \\
\qquad\qquad\qquad\quad \underline{\textbf{else}}\ \mathsf{root}(t) + \mathsf{sum}(\mathsf{left}(t), s \circ \langle \mathsf{right}(t)\rangle) \\
\quad \underline{\textbf{fi}}
\end{array}
$$

Dazu wählen wir die Abstiegsfunktion $h(t, s)$ folgendermaßen:

$$h(t, s) = 2 * \sum_i hs(s[i]) + \#s + 2 * hs(t)$$

Hinweis: Damit die Hilfsfunktion jeweils für die nächst tiefer verschachtelten rekursiven Aufrufe um 1 kleiner wird, wird in der Hilfsfunktion die Anzahl aller Knoten (Knoten der Bäume in der Sequenz s und die Knoten des Baumes t) gegenüber der Länge der Sequenz mit dem Faktor 2 gewichtet.

In der Abstiegsfunktion werden die folgenden Hilfsfunktionen benutzt:

hs: Tree $\alpha \to$ Nat	Anzahl der Knoten
$\_[i]$: Seq α, Nat $\to \alpha$	i-tes Element der Sequenz

Für die Hilfsfunktionen gelten die folgenden Gesetze:

$$hs(\mathsf{etree}) = 0$$

$$hs(\mathsf{cons}(l, w, r)) = hs(l) + hs(r) + 1$$

$$s[1] = \mathsf{first}(s)$$

$$s[n + 1] = \mathsf{rest}(s)[n] \Longrightarrow n > 0$$

Die Terminierung wird mittels der streng-monoton fallenenden Abstiegsfunktion h gezeigt:

- <u>Induktionsanfang:</u> Zu zeigen ist:

  $h(t, s) = 0 \implies \delta(\mathsf{sum}(t, s))$ ($\delta(\mathsf{sum}(t, s))$ bedeutet, dass $\mathsf{sum}(t, s)$ definiert ist.)

  Wir bekommen die Ableitung

$$
\begin{gathered}
h(t,s) = 0 \\
\implies \{\text{Def. von } h: \; 2 * \sum_i hs(s[i]) + \#s + 2 * hs(t) \\
2 * \sum_i hs(s[i]) = 0, \;\; \#s = 0 \text{ und } 2 * hs(t) = 0 \\
\implies s = \langle\rangle \;\wedge\; t = \mathsf{etree} \\
\mathsf{sum}(\mathsf{etree}, \langle\rangle) = 0 \quad (\mathsf{sum}(t,s) \text{ ist definiert})
\end{gathered}
$$

- <u>Induktionschritt:</u> Sei $n \in \mathbb{N}$, wir zeigen

  $\mathsf{sum}(t, s)$ ist definiert für alle t, s mit $h(t, s) = n$ *(Ind. Ann.)*
  dann gilt
  $\mathsf{sum}(t, s)$ ist definiert für $h(t, s) = n + 1$

  Beweis durch Fallunterscheidung (Fälle ergeben sich aus der Def. von sum.) :

  <u>1. Fall:</u> $t = \mathsf{etree}$ und $s \neq \langle\rangle$:

  Laut Def. von sum ergibt sich für obige Parameter:

$$
\mathsf{sum}(\mathsf{etree}, s) = \mathsf{sum}(\mathsf{first}(s), \mathsf{rest}(s))
$$

  Für die Abstiegsfunktion des ersten Parametertupels (etree, s) gilt:

$$
\begin{aligned}
h(\mathsf{etree}, s) &= 2 * \textstyle\sum_i hs(s[i]) + \#s + 2 * hs(\mathsf{etree}) \\
&\overset{(*)}{=} 2 * \textstyle\sum_i hs(s[i]) + \#s \\
&\overset{!}{=} n + 1
\end{aligned}
$$

$$
\left\{ \text{ Def. von } hs: \;\; hs(\mathsf{etree}) = 0 \right\} \qquad (*)
$$

  Für die Abstiegsfunktion des zweiten Parametertupels $(\mathsf{first}(s), \mathsf{rest}(s))$ gilt:

$$
\begin{aligned}
h(\mathsf{first}(s), \mathsf{rest}(s)) &= 2 * \textstyle\sum_i hs(\mathsf{rest}(s)[i]) + \#\mathsf{rest}(s) + 2 * hs(\mathsf{first}(s)) \\
&= 2 * \textstyle\sum_i hs(\mathsf{rest}(s)[i]) + \#s - 1 + 2 * hs(\mathsf{first}(s)) \\
&\overset{(**)}{=} 2 * \textstyle\sum_i hs((s)[i]) + \#s - 1 \\
&= n
\end{aligned}
$$

$$
\left\{ \text{ Def. von } \_[\_]: \quad 2 * \textstyle\sum_i hs(\mathsf{rest}(s)[i]) + 2 * hs(\mathsf{first}(s)) = 2 * \textstyle\sum_i hs((s)[i]) \right\} \qquad (**)
$$

Damit ergibt sich für die Abstiegsfunktionen der beiden Parameterpaare folgender Zusammenhang:

$$h(\mathsf{etree}, s) = h(\mathsf{first}(s), \mathsf{rest}(s)) + 1$$

Damit ist $\mathsf{sum}(t, s)$ in diesem Fall definiert.

<u>2. Fall:</u> $t \neq \mathsf{etree}$:

Es ergibt sich für sum:

$$\mathsf{sum}(t, s) = \mathsf{root}(t) + \mathsf{sum}\Big(\mathsf{left}(t),\ s \circ \langle\mathsf{right}(t)\rangle\Big)$$

Für die Abstiegsfunktion ergibt sich für das Parametertupel (t, s):

$$h(t, s) = 2 * \sum_i hs(s[i]) + \#s + 2 * hs(t) \overset{!}{=} n + 1$$

Für den rekursiven Aufruf ergibt sich für die Abstiegsfunktion h:

$$\begin{aligned}
h\Big(\mathsf{left}(t),\ s \circ \langle\mathsf{right}(t)\rangle\Big) &= 2 * \textstyle\sum_i hs\Big((s \circ \langle\mathsf{right}(t)\rangle)[i]\Big) + \#\Big(s \circ \langle\mathsf{right}(t)\rangle\Big) + 2 * hs(\mathsf{left}(t)) \\
&= 2 * \textstyle\sum_i hs(s[i]) + 2 * hs(\mathsf{right}(t)) + \#s + 1 + 2 * hs(\mathsf{left}(t)) \\
&= 2 * \textstyle\sum_i hs(s[i]) + \#s + 2 * \Big(hs(\mathsf{right}(t)) + 1 + hs(\mathsf{left}(t))\Big) - 1 \\
&\overset{(***)}{=} 2 * \textstyle\sum_i hs(s[i]) + \#s + 2 * hs(t) - 1 \\
&= n
\end{aligned}$$

$$\Big\{ \text{ Def. von } hs: \ hs(t) = hs(\mathsf{right}(t)) + 1 + hs(\mathsf{left}(t)) \Big\} \qquad (* * *)$$

Somit gilt:

$$h(t, s) = h\Big(\mathsf{left}(t), s(\langle\mathsf{right}(t)\rangle\Big) + 1$$

und damit ist $(\mathsf{sum}(t, s)$ ist definiert$)$ definiert.

Aufgabe 2 Sieb des Eratosthenes [LÖSUNG]

(a) Die Funktion mod liefert den Rest der Division $a \div b$.

```
fct mod = (a, b: Nat) Nat:
    if (a < b) then a
               else mod(a – b, b)
    fi
```

(b) Die Funktion sieb liefert eine Sequenz aller derjenigen Zahlen der Sequenz s, die *kein* Vielfaches der Zahl n, $n > 0$, sind. (Beachte: 0 ist Vielfaches jeder Zahl n.)

```
fct sieb = (n: Nat, s: Seq Nat) Seq Nat:
   if  iseseq(s)  then  ⟨⟩
                  else  if  (iszero( mod(first(s), n)) ))  ∨  ( iszero(first(s)))
                             then  sieb(n, rest(s))
                             else  first(s) ∘ sieb(n, rest(s))
                        fi
   fi
```

(c) Ein Aufruf der Funktion eratosthenes mit dem Argument $s = \langle 1, 2, 3, 4, 5, \ldots, n\rangle$ liefert eine Sequenz aller in s enthaltener Primzahlen. Dabei wird die Funktion sieb benutzt, um alle Vielfachen des jeweils ersten Elements der aktuellen Sequenz zu entfernen. Das erste Element selbst ist immer eine Primzahl (Spezialfall 0 bzw. 1) und wird jeweils in die Ergebnis-Sequenz aufgenommen.

```
fct eratosthenes = (s: Seq Nat) Seq Nat:
   if  iseseq(s)  then  ⟨⟩
                  else  if  ( iszero(first(s))  ∨  (first(s) = succ(0)) )
                             then  eratosthenes(rest(s))
                             else  first(s) ∘ eratosthenes(sieb(first(s), s)))
                        fi
   fi
```