Technische Universität München
Fakultät für Informatik
Dr. H. Ehler, Dr. M. Pizka,
W. Prenninger, Dr. M. Schneider

WS 2001
Lösungsvorschläge zu Blatt 12
31. Januar 2002

Übungen zu Einführung in die Informatik I

Aufgabe 48 Terminierungsbeweise für rekursive Funktionsdeklarationen (Lösungsvorschlag)

a) Wir definieren die Abstiegsfunktion $h : \mathbb{N} \to \mathbb{N}$ mit

$$h(x) = x$$

Zu zeigen:

(i) $h(x) = 0 \Rightarrow$ fac(x)$\neq\perp$ {IA – Induktionsanfang}

(ii) unter der Voraussetzung fac(x)$\neq\perp$ für $h(x) \leq k$:
$h(x) \leq k+1 \Rightarrow$ fac(x)$\neq\perp$ {IS – Induktionsschluß}

Beweis durch Induktion über k:

(i) k = 0
Aus der Definition von $h(x)$ und fac ergibt sich unmittelbar:
$h(x) = k = 0 \Rightarrow x = 0 \Rightarrow \text{fac}(x) = 1 \neq \perp$

(ii) $k \to k+1$
Für alle $x \in \mathbb{N}$, mit $x > 0$ gilt:

{fac$(x) \neq\perp$ gdw. rek. Aufruf in Rumpf terminiert}		$h(x-1)$
{Def. von h}	$=$	$x-1$
{$x \in \mathbb{N}$}	$<$	x
{Def. von h}	$=$	$h(x)$
{linke Seite IS}	$\leq$	$k+1$
	$\Rightarrow$	$h(x-1) \leq k$
{nach I.-Voraus.}	$\Rightarrow$	fac$(x-1) \neq\perp$

Mit der Def. von fac und der Voraussetzung $x > 0$ folgt: fac(x)=x*fac(x-1) $\neq\perp$

b) Als Abstiegsfunktion wählen wir $h : \mathbb{N}^* \times \mathbb{N}^* \to \mathbb{N}$ mit:

$$h(a,b) := \begin{cases} 0 & \text{, falls a} \stackrel{?}{=} \text{empty} \vee \text{b} \stackrel{?}{=} \text{empty} \\ laenge(a) + laenge(b) & \text{, falls a} \neq \text{empty} \wedge \text{b} \neq \text{empty} \end{cases}$$

Der Induktionsanfang ergibt sich aus der Definition von schnitt. Erfolgt kein rekursiver Aufruf, d.h. ist $a \stackrel{?}{=}$ empty $\vee$ $b \stackrel{?}{=}$ empty erfüllt, so ist $h(a,b) = 0$ und schnitt(a, b) terminiert mit dem Ergebnis empty.

Im Falle eines rekursiven Aufrufs ist $a \stackrel{?}{=}$ empty $\vee$ $b \stackrel{?}{=}$ empty nicht erfüllt, $h(a,b) \neq 0$ und es sind drei Fälle zu betrachten:

(i) first(a) $\stackrel{?}{=}$ first(b)
rekursiver Aufruf schnitt(rest(a), rest(b)).

$$\begin{array}{rcl} h(rest(a), rest(b)) & = & \begin{cases} 0 & \text{, rest(a)} \stackrel{?}{=} \text{empty} \vee \text{rest(b)} \stackrel{?}{=} \text{empty} \\ laenge(a) + laenge(b) - 2 & \text{, rest(a)} \neq \text{empty} \wedge \text{rest(b)} \neq \text{empty} \end{cases} \\ & < & h(a,b) \end{array}$$

(ii) first(a) < first(b)
rekursiver Aufruf schnitt(rest(a), b)

$$\begin{array}{rcl} h(rest(a), b) & = & \begin{cases} 0 & \text{, rest(a)} \stackrel{?}{=} \text{empty} \vee \text{b} \stackrel{?}{=} \text{empty} \\ laenge(a) + laenge(b) - 1 & \text{, rest(a)} \neq \text{empty} \wedge \text{b} \neq \text{empty} \end{cases} \\ & < & h(a,b) \end{array}$$

(iii) first(a) > first(b)
rekursiver Aufruf schnitt(a, rest(b))

$$\begin{array}{rcl} h(a, rest(b)) & = & \begin{cases} 0 & \text{, a} \stackrel{?}{=} \text{empty} \vee \text{rest(b)} \stackrel{?}{=} \text{empty} \\ laenge(a) + laenge(b) - 1 & \text{, a} \neq \text{empty} \wedge \text{rest(b)} \neq \text{empty} \end{cases} \\ & < & h(a,b) \end{array}$$

Im Fall eines rekursiven Aufrufes tritt also entweder sofort der Terminierungsfall ein, falls $h(rest(a), rest(b)) = 0$, $h(a, rest(b)) = 0$ oder $h(rest(a), b) = 0$ oder h verringert sich um 2 bzw. 1. Jede absteigende Folge in $\mathbb{N}$ ist aufgrund der Fundiertheit von $\mathbb{N}$ endlich und damit terminiert die Funktion schnitt nach endlich vielen Schritten.

Aufgabe 49 Terminierung rekursiver Funktionen (Lösungsvorschlag)

a) Bei jedem rekursiven Aufruf von f wird der Abstand zwischen erstem und zweitem Argument echt kleiner.

Diese Beobachtung fomalisieren wir in einer Abstiegsfunktion h. Wir definieren $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ durch

$$h(x,y) = \begin{cases} y - x, & \text{falls} \quad x < y \\ 0, & \text{falls} \quad x \geq y \end{cases}$$

für alle $x, y \in \mathbb{N}$.

Nun beweisen wir, daß h tatsächlich eine Abstiegsfunktion für f ist.

(i) Für alle $x, y \in \mathbb{N}$ gilt:

$$\begin{array}{rcl} & & h(x,y) = 0 \\ \{\text{Def. von } h\} & \equiv & x \geq y \\ \{\text{Def. von f}\} & \Rightarrow & f(x,y) = 1 \vee f(x,y) = x \\ \{x \in \mathbb{N}\} & \Rightarrow & f(x,y) \neq \perp \end{array}$$

(ii) Für alle $x, y \in \mathbb{N}$ mit $x < y$ gilt:

$$\begin{array}{rcl} & & h(x, (x+y) \div 2) \\ \{\text{Def. von h}\} & = & \begin{cases} (x+y) \div 2 - x, & \text{falls} \quad x < (x+y) \div 2 \\ 0, & \text{falls} \quad x \geq (x+y) \div 2 \end{cases} \\ \{\text{aus } x < y \text{ folgt } x + y < 2 * y & < & y - x \\ \text{also } (x+y) \div 2 < y \text{ und damit} & & \\ (x+y) \div 2 - x < y - x; \text{ aus} & & \\ x < y \text{ folgt außerdem } 0 < y - x\} & & \\ \{\text{Def. von } h, \text{ Vor. } x < y\} & = & h(x,y) \end{array}$$

(iii) Für alle $x,y \in \mathbb{N}$ mit $x < y$ gilt:

$$
\begin{array}{lcl}
 & & h(1+(x+y)\div 2, y) \\
\{\text{Def. von } h\} & = & \begin{cases} y-1-(x+y)\div 2, & \text{falls } 1+(x+y)\div 2 < y \\ 0, & \text{falls } 1+(x+y)\div 2 \geq y \end{cases} \\
\{\text{aus } x<y \text{ folgt } 2*x < x+y & < & y-x \\
\text{also } x < 1+(x+y)\div 2 \text{ und damit} & & \\
y-1-1(x+y)\div 2 < y-x\text{; aus} & & \\
x<y \text{ folgt außerdem } 0 < y-x\} & & \\
\{\text{Def. von } h\text{, Vor. } x<y\} & = & h(x,y)
\end{array}
$$

Damit ist gezeigt, daß h eine Abstiegsfunktion für f ist.

Aufgabe 50 Einbettung (Lösungsvorschlag)

a) **Minimumsuche:**

```
minimum :: [Int] -> Int
minimum [] = undefined
minimum s  = minimum2 (head s) (tail s)

minimum2 :: Int -> [Int] -> Int
minimum2 x [] = x
minimum2 x (y:s) | x<y = minimum2 x s
           | otherwise = minimum2 y s
```

b) **Berechnung der ersten n Primzahlzwillinge:**

```
isprim :: Int -> Bool
isprim n = (n >=2) && (not (isdiv (n/2) n))

isdiv :: Int -> Int -> Bool
isdiv k n | k <= 1 = False
          | otherwise = (mod n k == 0) || (isdiv (k-1) n)

goldbach :: Int -> [(Int, Int)]
goldbach n = zwillinge 2 n []

zwillinge :: Int -> Int -> [(Int, Int)] -> [(Int, Int)]
zwillinge z 0 ll = ll
zwillinge z n ll | iszwilling  = zwillinge (z+2) (n-1) ((z, z+2): ll)
                 | otherwise = zwillinge (z+1) n ll
                   where iszwilling = isprim z && isprim (z+2)
```

c) **Implementierung der Goferfunktion** `concat`:

```
reduziere :: [[a]] -> [a]
reduziere x = verbinde [] x

verbinde :: [a] -> [[a]] -> [a]
verbinde l [] = l
verbinde l (x:ll) = verbinde (l++x) ll
```

Aufgabe 51 Zuweisungen (Lösungsvorschlag)

a) Zuweisungen bewirken Zustandsänderungen. Zustände sind Belegungen, die den Programmvariablen Werte zuordnen. $\perp$ ist der undefinierte Zustand. Für das gegebene Programm haben wir folgende Zustandsänderungen:

Zust. vor Zuweis.	Zuweisung	Zustand nach Zuweisung
$\sigma_0 \neq \perp$	$x := 4;$	$\sigma_1 = \sigma_0[4/x]$
σ_1	$y := 2;$	$\sigma_2 = \sigma_1[2/y] = \sigma_0[4/x][2/y]$
σ_2	$y := x+1;$	$\sigma_3 = \sigma_2[5/y] = \sigma_0[4/x][5/y]$
σ_3	$x := y+1;$	$\sigma_4 = \sigma_3[6/x] = \sigma_0[6/x][5/y]$
σ_4	$x := x*x;$	$\sigma_5 = \sigma_4[36/x] = \sigma_0[36/x][5/y]$

Schreiben wir x_{neu}, y_{neu} für $\sigma_5(x)$, $\sigma_5(y)$, so erhalten wir $x_{neu} = 36$, $y_{neu} = 5$.

b) Initialzustand für alle drei Anweisungen sei $\sigma_0 \neq \perp$.

(i)

$\sigma_0 \neq \perp$	$x := y+1;$	$\sigma_1 = \sigma_0[\sigma_0(y)+1/x]$
σ_1	$y := x+1;$	$\sigma_2 = \sigma_1[\sigma_1(x)+1/y] = \sigma_1[\sigma_0(y)+2/y] =$ $= \sigma_0[\sigma_0(y)+1/x][\sigma_0(y)+2/y]$

Schreiben wir x_{alt}, y_{alt} für $\sigma_0(x)$, $\sigma_0(y)$ und x_{neu}, y_{neu} für $\sigma_2(x)$, $\sigma_2(y)$, so gilt $x_{neu} = y_{alt}+1$, $y_{neu} = y_{alt}+2$.

(ii)

$\sigma_0 \neq \perp$ $\quad (x,y) := (y+1, x+1);$ $\quad \sigma_1 = \sigma_0[\sigma_0(y)+1/x][\sigma_0(x)+1/y]$

D.h. es gilt $x_{neu} = y_{alt}+1$, $y_{neu} = x_{alt}+1$.

(iii)

$\sigma_0 \neq \perp$	$y := x+1;$	$\sigma_1 = \sigma_0[\sigma_0(x)+1/y]$
σ_1	$x := y+1;$	$\sigma_2 = \sigma_1[\sigma_1(y)+1/x] = \sigma_1[\sigma_0(x)+2/x] =$ $= \sigma_0[\sigma_0(x)+1/y][\sigma_0(x)+2/x]$

D.h. es gilt $x_{neu} = x_{alt}+2$, $y_{neu} = x_{alt}+1$.

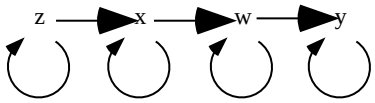
Die Zustandänderung der kollektiven Zuweisung läßt sich nur mit einer zusätzlichen Hilfsvariablen realisieren:

$\sigma_0 \neq \perp$	$h := x;$	$\sigma_1 = \sigma_0[\sigma_0(x)/h]$
σ_1	$x := y+1;$	$\sigma_2 = \sigma_1[\sigma_0(y)+1/x] = \sigma_0[\sigma_0(x)/h][\sigma_0(y)+1/x]$
σ_2	$y := h+1;$	$\sigma_3 = \sigma_2[\sigma_2(h)+1/y] = \sigma_0[\sigma_0(x)/h][\sigma_0(y)+1/x][\sigma_0(x)+1/y]$

Streng gesehen ist auch diese Zustandsänderung nicht identisch mit der der kollektiven Zuweisung, da das σ_3 von hier nicht mit σ_1 in (ii) übereinstimmt. Hier wird nämlich σ_0 auch an der Stelle der Hilfsvariable h verändert.

c) $z := z + x; \quad x := x + w; \quad w := w + y; \quad y := y + 1;$

Die Reihenfolge der Zuweisungen an *z, x, w, y* ist die einzige, die ohne Hilfsvariablen auskommt. Man findet diese Reihenfolge über den folgenden Graph, der angibt welche neuen Variablenwerte von welchen alten abhängig sind.



Aufgabe 52 Hornerschema/Pascal (Lösungsvorschlag)

a) b) c) d) e)

```
program aufgabe52 (Input, Output);

{Variablendeklaration}
var x, yoh, ymh : Integer;

{Hauptprogramm}
begin

   x := 4;

   {Berechnung ohne Hornerschema}
   yoh := 2*x*x*x*x*x - x*x*x*x + 4*x*x*x - 3*x*x + 2;

   {Berechnung mit Hornerschema}
   ymh := 0;
   ymh := ymh*x + 2;
   ymh := ymh*x - 1;
   ymh := ymh*x + 4;
   ymh := ymh*x - 3;
   ymh := ymh*x;
   ymh := ymh*x + 2;

   {Ausgabe der Ergebnisse}
   WriteLn ('Ergebnis ohne Hornerschema: ', yoh);
   WriteLn ('Ergebnis mit Hornerschema : ', ymh);
   WriteLn;

end.
```

f) Anzahl der arithmetischen Elementaroperationen:

- ohne Hornerschema 17,
- mit Hornerschema 11.