Rekursive Funktionen und Berechenbarkeit

Seien $\mathbf{x, y, a, b, x_i, y_i} \in \mathbb{N}_0$ mit $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$; in Pascal-Programmen also vom Typ *integer*.

Unter der Produktmenge **A** × **B** zweier Mengen **A** und **B** verstehen wir die Menge aller geordneten Paare (ein solches Paar läßt sich als zweidimensionaler Vektor auffassen), deren erste Komponente aus **A** und deren zweite Komponente aus **B** ist:

$\mathbf{A = \{a_1, a_2, \ldots, a_n\}}$
$\mathbf{B = \{b_1, b_2, \ldots, b_m\}}$

$\mathbf{A \times B := \{ (a_i, b_j) \mid a_i \in A \text{ und } b_j \in B \}}$

Beachte: die Produktmenge A × B enthält $n \cdot m$ Elemente.

1. Elementare Funktionen („basic functions")

a) Nachfolgerfunktion:

$$\mathbf{S:}\quad \mathbb{N}_0 \to \mathbb{N}_0, \quad x \to \mathrm{succ}(x)$$

$$\mathbf{S(x) := succ(x) = x + 1}$$

b) Nullfunktion ("zero function"):

$$\mathbf{N:}\quad \mathbb{N}_0 \to \mathbb{N}_0, \quad x \to 0$$

$$\mathbf{N(x) := 0}$$

c) Projektionsfunktion ("generalized identity function")

$$\mathbf{U^{(2)}_i :}\quad \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0, \quad (x_1, x_2) \to x_i$$

$$\mathbf{U^{(2)}_i(x_1, x_2) = x_i}$$

allgemein:

$$\mathbf{U^{(n)}_i :}\quad \mathbb{N}_0^{\,n} \to \mathbb{N}_0, \quad (x_1, x_2, \ldots, x_n) \to x_i$$

$$\mathbf{U^{(n)}_i(x_1, x_2, \ldots, x_n) = x_i}$$

2. Primitiv-rekursive Funktionen („primitive recursive functions")

Definition:
Eine Funktion $\mathbf{r} : \mathbb{N}_0^{\,\mathbf{n}} \to \mathbb{N}_0$ heißt primitiv-rekursiv genau dann, wenn sie gemäß folgender in a), b), c) gemachten Vorgaben gebildet wird (zunächst: Beschränkung auf $\mathbf{n \le 2}$):

a) Die **Grundfunktionen** und **Verkettungen** aus diesen sind primitiv-rekursiv.

b) Verkettung („composition"):
Falls **g:** $\mathbb{N}_0^{\,2} \rightarrow \mathbb{N}_0$
und $\mathbf{f_i}$**:** $\mathbb{N}_0 \rightarrow \mathbb{N}_0$, $i \in \{1, 2\}$
primitiv-rekursiv sind, dann ist auch die Funktion

$$\mathbf{r}: \mathbb{N}_0 \rightarrow \mathbb{N}_0$$

mit $\mathbf{r(x) := g(f_1(x), f_2(x))}$ primitiv rekursiv.

c) Primitive Rekursion („primitive recursion")
Falls **g:** $\mathbb{N}_0 \rightarrow \mathbb{N}_0$
und **h :** $\mathbb{N}_0^{\,3} \rightarrow \mathbb{N}_0$
primitiv-rekursiv sind, dann ist auch die Funktion

$$\mathbf{r}: \mathbb{N}_0^{\,2} \rightarrow \mathbb{N}_0$$

mit $\mathbf{r(x,0) := g(x)}$
und $\mathbf{r(x,y) := h(x, y-1, r(x, y-1))}$, falls $y \geq 1$
primitiv rekursiv.

Beispiele primitiv-rekursiver Funktionen:

a) $\mathbf{f(x) = 1}$ ist primitiv-rekursiv, denn

$\mathbf{f(x) = 1 = 0+1 = S(0) = S(N(x))}$

(hier: f(x) als Verkettung von Grundfunktionen)

b) **Sum :** $\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$
$\mathbf{(x,y) \rightarrow x + y}$

oder als Funktionsgleichung geschrieben:

$\mathbf{Sum(x, y) = x + y}$

ist primitiv-rekursiv, denn

$$\begin{aligned}\mathbf{Sum(x, 0)} &= \mathbf{x}\\ \mathbf{Sum(x, y) = x + y} &= \mathbf{[x + (y-1)] + 1}\\ &= \mathbf{S[x + (y-1)]}\\ &= \mathbf{S(Sum(x, y-1))} \quad \text{falls } y \geq 1.\end{aligned}$$

Damit ist **Sum** primitiv-rekursiv nach Definition c)
mit $\mathbf{g(x)=x}$
und $\mathbf{h(x, y-1, Sum(x, y-1)) = f(U^{(3)}{}_3(x, y-1, Sum(x, y-1))}$
$= \mathbf{f(Sum(x, y-1))}$
$= \mathbf{S(Sum(x, y-1))}$ mit S = f

c) **Prod :** $\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$
$\mathbf{(x,y) \rightarrow x \cdot y}$

oder als Funktionsgleichung geschrieben:

$\mathbf{Prod(x, y) = x \cdot y}$

ist primitiv-rekursiv, denn

$$\begin{aligned}
\mathbf{Prod(x,\ 0)} &= \mathbf{0} \\
\mathbf{Prod(x,\ y)} &= \mathbf{x \cdot y = x \cdot (y-1) + x} \\
&= \mathbf{Prod(x,\ y-1) + x} \\
&= \mathbf{Sum(Prod(x,\ y-1),\ x)} \\
&= \mathbf{Sum(Prod(x,\ y-1),\ U^{(2)}{}_1(x\ ,\ y\ ))} \\
&= \mathbf{Sum(U^{(2)}{}_1(x\ ,\ y\ ),\ Prod(x,\ y-1))} \quad , \ \text{falls } y \geq 1
\end{aligned}$$

Damit ist **Prod** ebenfalls primitiv-rekursiv nach Definition c).

d) Pot : $\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$

$\mathbf{(x,y)} \rightarrow \mathbf{x^y}$

oder als Funktionsgleichung geschrieben:

$\mathbf{Pot(x,\ y) = x^y}$

ist primitiv-rekursiv, denn

$\mathbf{Pot(x,\ 0) = x^0 = 1}$

$\mathbf{Pot(x,\ y) = x^y = x^{y-1} \cdot x =}$ . . . . (Übungsaufgabe, analog zu Beispiel c))

e) Fact : $\mathbb{N}_0 \rightarrow \mathbb{N}_0$

$\mathbf{x} \rightarrow \mathbf{x!}$

oder als Funktionsgleichung geschrieben:

$\mathbf{Fact(x) = x!}$

ist primitiv-rekursiv, denn

$\mathbf{Fact(0) = 0! = 1}$

$\mathbf{Fact(x) = x! = x \cdot (x-1)! =}$ . . . . (Übungsaufgabe, analog zu Beispiel c))

Pascal-Programm, welches obenstehende primitiv-rekursiven Funktionen jeweils als ***function*** enthält; beachte: da einige der Funktionen zweckmäßigerweise andere aufrufen, müssen im Programmtext solche aufgerufenen Funktionen *vor* der rufenden Funktion stehen.

Das Pascalprogramm hat also folgenden Aufbau:

Eingabe von x, y

Auswahl der Operation (Funktion), die auf x, y wirkt

Ausgabe des Ergebnisses

Pascal-Programm:

```
program RecFunc;
uses crt;

var
  x, y, i: integer;
  ans: char;

function S(a: integer): integer;
begin
  S := a + 1;
end;

function N(a: integer): integer;
begin
  N := 0;
end;

function U(a, b, i: integer): integer;
begin
  if i=1 then U := a
         else U := b
  //else U := nil;
end;

function Sum(a, b: integer): integer;
begin
  if b = 0 then Sum := a
           else Sum := S(Sum(a, b-1))
  //else Sum := nil;
end;

function Prod(a, b: integer): integer;
begin
  if b = 0 then Prod := 0
           else Prod := Sum(Prod(a,b-1), U(a, b, 1))
  //else Prod := nil;
end;

function Pot(a, b: integer): integer;
begin
  //I was here....
end;

begin
  Writeln('1 -> X + 1');
  Writeln('2 -> X = 0');
  Writeln('3 -> U(2) i');
  Writeln('4 -> Summe');
  Writeln('5 -> Produkt');
  Write('Zahl von 1 bis 5... ');
  readln(ans);
  Write('x = '); Readln(x);
  Write('y = '); Readln(y);
  if (ans = '1') then Writeln(S(x));
  if (ans = '2') then Writeln(N(x));
  if (ans = '3') then Writeln(U(x, y, 2));
  if (ans = '4') then Writeln(Sum(x, y));
  if (ans = '5') then Writeln(Prod(x, y));
  repeat until keypressed;
  while not keypressed do
end.
```

Aufgabe: Zeige, daß folgende Funktionen primitiv-rekursiv sind:

f) Diff : $\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$

$(x,y) \rightarrow \mathrm{Diff}(x,y)$ mit $\mathrm{Diff}(x,y) := x - y$ falls $x \geq y$
$\mathrm{Diff}(x,y) := 0$ falls $x < y$

g) Pred: $\mathbb{N}_0 \rightarrow \mathbb{N}_0$

$x \rightarrow \mathrm{Pred}(x)$ mit $\mathrm{Pred}(x) := x - 1$ falls $x \geq 1$
$\mathrm{Pred}(x) := 0$ falls $x = 0$

h) Abs: $\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$

$(x,y) \rightarrow |x - y|$

i) Man mache sich folgende Äquivalenz klar:

$$x \geq y \quad \Leftrightarrow \quad \mathrm{Diff}(y,x) = 0$$

3. Partiell-rekursive Funktionen („partial recursive functions“)

Wir erweitern die Klasse primitiv-rekursiver Funktionen, indem wir folgende Vorschrift zur Erzeugung von Funktionen formulieren:

„Minimalization“ (auch: **μ-operator**)

Für eine gegebene Funktion f(y,x) definieren wir die Funktion h wie folgt:

$$h(x) := \min \{ y \mid f(y,x) = 0 \}$$

lies: „das Minimum aller Werte von y, für die gilt: f(y,x)=0“

Beispiel:

Die Funktion
$h(x) := [x/2] = x \text{ DIV } 2$ ([. . .] = Gauß-Klammer)
läßt sich durch einen geeigneten Minimalisierungsprozeß definieren.

$$\begin{aligned} h(x) &= \min \{ y \mid 2(y+1) > x \} \\ &= \min \{ y \mid 2y + 2 > x \} \\ &= \min \{ y \mid 2y + 2 \geq x + 1 \} \\ &= \min \{ y \mid 2y + 1 \geq x \} \\ &= \min \{ y \mid (2y + 1) - x \geq 0 \} \\ &= \min \{ y \mid \mathrm{Diff}(x, 2y + 1) = 0 \} \end{aligned}$$

Flußdiagramm zur Berechnung mittels des μ-Operators definierter Funktionen: