GTI

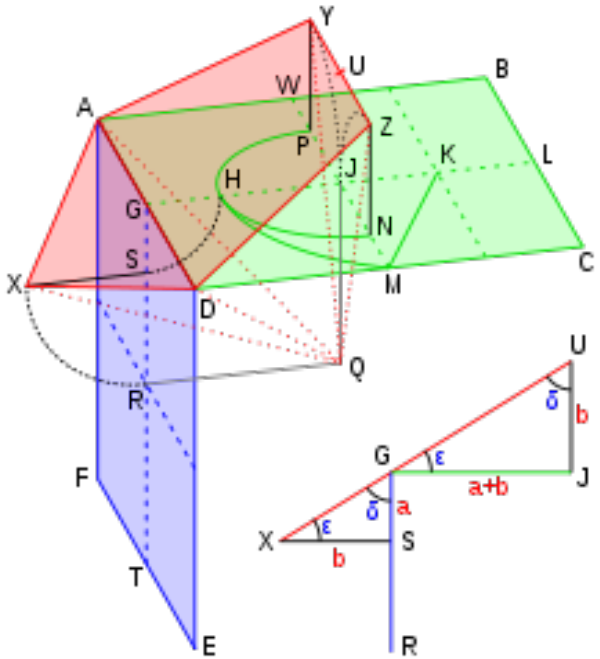
μ-rekursive Funktionen

Hannes Diener

ENC B-0123,
diener@math.uni-siegen.de

20. Juni – 2. Juli

μ-rekursive Funktionen



Kommen wir
als nächstes zu unserem dritten Ansatz zur Berechenbarkeit.
Diesmal werden wir eine rein mathematische Charakterisierung der Berechenbarkeit kennenlernen.

Die Idee ist, das wir *induktiv* eine Klasse von Funktionen aufbauen, die wir als berechenbar ansehen. D.h.

- wir identifizieren einige Funktionen, sogenannten *Basisfunktionen* die auf alle Fälle berechenbar sein sollen und
- erklären mit welchen Konstruktionen wir berechenbare Funktionen zu neuen berechenbaren Funktionen kombinieren können.

Indem wir die zweite Regel iterieren können wir also immer komplexere Funktionen basteln.

Genauer:

Definition

Die Basisfunktionen sind

1. *Die Nullfunktion* $N = \lambda n.0$ *ist eine Basisfunktion.*
2. *Die Nachfolgefunktion* $S = \lambda n.n + 1$ *ist eine Basisfunktion.*
3. *Die Projektionen* U_i^k *auf die i-te Koordinate, also* $U(n_1, \ldots, n_k) = n_i$ *sind Basisfunktionen.*

Insbesondere ist auch die Identität Basisfunktion (U_1^1)

Definition

Die primitiv rekursiven Funktionen sind definiert durch

1. *Jede Basisfunktion ist primitiv rekursiv.*
2. *Sind* $h : \mathbb{N}^k \to \mathbb{N}$ *und* $f_1, \ldots, f_k : \mathbb{N}^\ell \to \mathbb{N}$, *so auch die Funktion* $\mathbb{N}^\ell \to \mathbb{N}$ *definiert durch*

$$h\left(f_1(n_1, \ldots, n_\ell), \ldots, f_k(n_1, \ldots, n_\ell)\right)$$

3. *Seien* $h : \mathbb{N}^{k+2} \to \mathbb{N}$, $f : \mathbb{N}^{k+1} \to \mathbb{N}$ *und* $g : \mathbb{N}^k \to \mathbb{N}$ *Funktionen, so daß*

$$\begin{aligned} f(0, y_1, \ldots, y_k) &= g(y_1, \ldots, y_k) \\ f(n+1, y_1, \ldots, y_k) &= h(n, f(n, y_1, \ldots, y_k), y_1, \ldots, y_k) \ . \end{aligned}$$

Sind h, g *primitiv rekursiv, so auch* f.

Da es sich um eine induktive Definition handelt bietet es sich also an, um zu zeigen, daß eine Funktion primitiv rekursiv ist sich einen Vorrat an nützlichenen primitiv rekursiven Funktionen anzulegen.

Ähnlich wie bei den LOOP-Programmen gilt auch hier

Beobachtung

Primitiv rekursive Funktionen sind total.

D.h. auch hier, das die primitiv rekursiven Funktionen nicht genau unserem intuitiven Begriff der Berechenbarkeit ausfüllen.

Aber zunächst Beispiele:

Beispiel

Die Addition $f_+(x, y) = x + y$ können wir kompliziert schreiben als:

$$\begin{aligned} f_+(0, y) &= y = U_1^1(y) \\ f_+(x+1, y) &= f_+(x, y) + 1 = S(f_+(x, y)) \\ &= S \circ U_2^3(x, f_+(x, y), y) \end{aligned}$$

D.h. f_+ geht durch primitive Rekursion aus g und h hervor, ist also

Satz

Die folgenden Funktionen sind primitiv-rekursiv:

(1) $\lambda x.k,\ k \in \mathbb{N}$

(2) $\lambda x, y.x + y$

(3) $\lambda x.k \cdot x,\ k \in \mathbb{N}$

(4) $\lambda x, y.x \cdot y$

(5) $\lambda x, y.x^y$

(6) $\lambda x.x^k,\ k \in \mathbb{N}$

(7) $\lambda x.k^x,\ k \in \mathbb{N}$

(8) $\mathrm{sg}, \overline{\mathrm{sg}}$, *wobei*

$$\mathrm{sg}(x) = \begin{cases} 0, & x = 0 \\ 1, & \textit{sonst} \end{cases}$$

$$\overline{\mathrm{sg}}(x) = \begin{cases} 1, & x = 0 \\ 0, & \textit{sonst} \end{cases}$$

(9) $\lambda x, y.x \cdot \mathrm{sg}(y)$

(10) $\lambda x, y.x \cdot \overline{\mathrm{sg}}(y)$

(11) $\lambda x, y.x \dot{-} y$, *wobei*

$$x \dot{-} y = \begin{cases} x - y, & x \geqslant y \\ 0, & \textit{sonst} \end{cases}$$

(12) $\lambda x, y.|x - y|$

(13) $\lambda x.\left\lfloor \frac{x}{2} \right\rfloor$

(14) $\lambda x.\lfloor \sqrt{x} \rfloor$

(15) $\lambda x, y.\max\{x, y\}$

(16) $\lambda x, y.\min\{x, y\}$

In vielen Fällen erzeugen wir neue Funktionen aus bekannten mittels Fallunterscheidungen. Wir wollen nun sehen, dass diese Operation nicht aus dem Bereich der primitiven Rekursion herausführt.

Satz (Definition durch Fallunterscheidung)

Seien $h, f_1, f_2 : \mathbb{N}^k \to \mathbb{N}$ *und sei* $f : \mathbb{N}^k \to \mathbb{N}$ *definiert durch*

$$f(x_1, \ldots, x_k) = \begin{cases} f_1(x_1, \ldots, x_k), & \text{falls } h(x_1, \ldots, x_k) = 0 \\ f_2(x_1, \ldots, x_k), & \text{falls } h(x_1, \ldots, x_k) \neq 0. \end{cases}$$

Dann ist f *primitiv rekursiv.*

Satz

Ist $f : \mathbb{N}^{k+1} \to \mathbb{N}$ *primitiv rekursiv, so auch die Iteration, d.h. die Funktion* $f^m(x_1, \ldots, x_k, y) : \mathbb{N}^{k+2} \to \mathbb{N}$ *definiert durch*

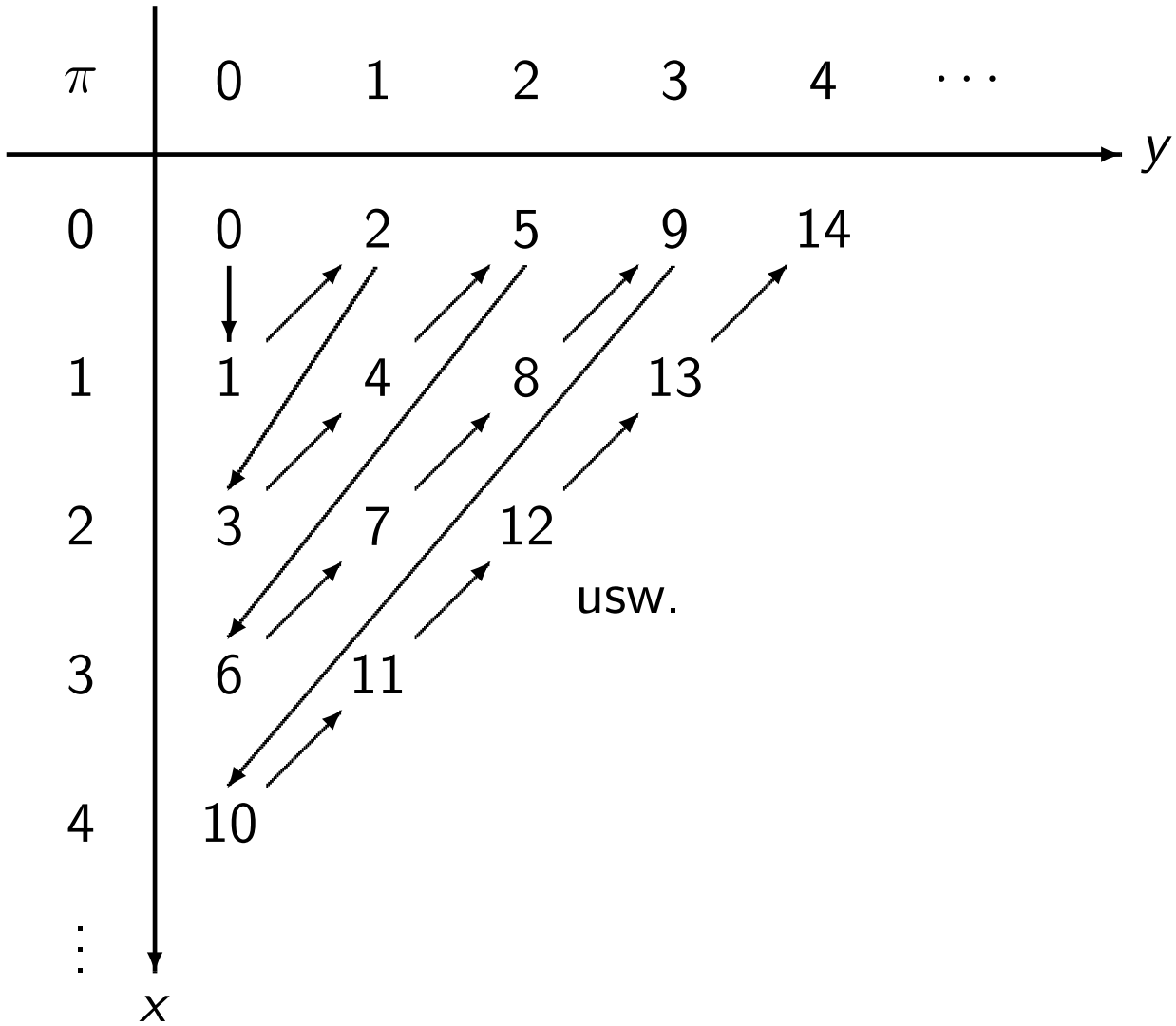
$$f^0(x_1, \ldots, x_k, y) = y$$
$$f^{m+1}(x_1, \ldots, x_k, y) = f(x_1, \ldots, x_k, f^m(x_1, \ldots, x_k, y))$$

Im Falle $k = 0$ *heißt das also,* $f^0(y) = y$ *und*

$$f^m(y) = \underbrace{f(f(\ldots(f(y))\ldots))}_{m\text{-mal}}$$

Wie wir wissen sind $\mathbb{N}^2$ und $\mathbb{N}$ gleichmächtig, d.h. wir können jedes Paar von zwei natürlichen Zahlen eindeutig als eine einzige Zahl kodieren. Wie wir gleich sehen ist dieser Übergang primitiv rekursiv!

Eine der bekanntesten Bijektionen $\mathbb{N} \to \mathbb{N}^2$ ist die folgende:



Wie man sieht, ist

$$\pi(x,0) = \sum_{i \leqslant x} i = \frac{1}{2}(x+1)x \ .$$

Wir erhalten somit, dass

$$\pi(x,y) = \frac{1}{2}(x+y)(x+y+1) + y \ . \tag{1}$$

Wie man leicht sieht ist π bijektiv. Also existiert die Umkehrfunktion π^{-1}. Für $i = 1, 2$ sei $\pi_i = U_i^2 \circ \pi^{-1}$. π_i heißt die Projektion auf die i-te Komponente.

Satz

π, π_1, π_2 *sind primitiv rekursiv.*

Diese Kodierungen wollen wir verwenden, um Listen fester Länge von natürlicher Zahlen zu kodieren. Sei hierfür induktiv die k-Tupel definiert

$$\langle x \rangle^1 = x$$

und

$$\langle x_1, x_2, \ldots, x_{k+1} \rangle^{k+1} = \pi \left(x_1, \langle x_2, \ldots, x_{k+1} \rangle^k \right)$$

Satz

Sei $k \in \mathbb{N}$ fest gewählt. Dann ist die Funktion

$$\lambda x_1, \ldots, x_k . \langle x_1, \ldots, x_k \rangle^k$$

bijektiv und primitiv rekursiv.
Ebenso sind die Funktionen $(\cdot)_i^k : \mathbb{N} \to \mathbb{N}$ definiert[1] *durch*

$$(\langle x_1, \ldots, x_k \rangle)_i^k = x_i$$

primitiv-rekursiv.

Ist die Länge k klar, so wollen wir auch $\langle x_1, x_2, \ldots, x_k \rangle$ an Stelle von $\langle x_1, x_2, \ldots, x_k \rangle^k$ schreiben.

[1] Existenz ist sichergestellt wegen der Bijektivität

Ähnlich können wir Listen von natürlicher Zahlen beliebiger Länge kodieren. Sei hierfür induktiv

$$\langle\rangle^* = \langle 0 \rangle^1$$

und

$$\langle x_1, x_2, \ldots, x_k \rangle^* = \left\langle k, x_1, \langle x_2, \ldots, x_k \rangle^* \right\rangle^3$$

Satz

Die Funktion $\mathrm{lth} : \mathbb{N} \to \mathbb{N}$ *einfach definiert durch* $(\cdot)_1^3$ *hat die Eigenschaft, daß*

$$\mathrm{lth}\left(\langle x_1, x_2, \ldots, x_k \rangle^*\right) = k \ .$$

Ebenso gibt es primitiv-rekursive Funktionen $(\cdot)_i^*$*, die die Eigenschaft haben, daß*

$$\left(\langle x_1, x_2, \ldots, x_k \rangle^*\right)_i^* = x_i \ ,$$

falls $1 \leqslant i \leqslant k$*.*

Grenzen der primitiven Rekursion: die Ackermannfunktion

Im folgenden wollen wir zeigen, daß der Begriff der primitiv-rekusiven Funktionen noch nicht ganz unserem intuitiven Begriff der Berechenbarkeit entspricht. Wir geben hierzu eine Funktion an, welche zwar intuitiv berechenbar ist, aber schneller wächst als jede primitiv-rekursive Funktion, und damit selber nicht primitiv-rekursiv sein kann.

„Die“[2] Ackermannfunktion ist definiert durch

$$B(0,x) = \begin{cases} 1, & \text{falls } x = 0 \\ 2, & \text{falls } x = 1 \\ x+2, & \text{sonst} \end{cases}$$

$$B(n+1,x) = B^x(n,1) = \underbrace{B(n,(B(n,\ldots B}_{x\text{-mal}}(n,1)\ldots))$$

Die Funktion $B = \lambda n.x.B(n,x)$ heißt *Ackermannfunktion*, $B_n = \lambda x.B(n,x)$ ihr *n-ter Zweig*.

[2] Es gibt mehrere Varianten dieser Funktion, wobei die Idee aber immer die gleiche ist. Ackermanns originale Definition von 1929 wird in der Literatur so gut wie überhaupt nicht mehr verwendet, da es einfachere Darstellungen gibt.

Lemma

1. *Für* $x > 1$ *ist* $B(0, x) = x + 2$.
2. *Für* $x > 0$ *ist* $B(1, x) = 2x$.
3. *Für* $x \in \mathbb{N}$ *ist* $B(2, x) = 2^x$.
4. *Für* $x > 0$ *ist* $B(3, x) = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\}\, x\text{-mal}$.

Die Ackermann-Funktion wächst fast unvorstellbar schnell. Hier einige der ersten Werte als Tabelle

$n \backslash x$	0	1	2	3	4	5	6
0	1	2	4	5	6	7	8
1	1	2	4	6	8	10	12
2	1	2	4	8	16	32	64
3	1	2	2^2	2^{2^2}	$2^{2^{2^2}}$	2^{65536}	$2^{2^{65536}}$

Schon $B(3,5)$ lässt sich nur gerade noch mit Hilfe eines Computers bestimmen. Bei $B(3,6)$ ist das so gut wie unmöglich.

Satz

Zu jeder Stelligkeit n und jeder n-stelligen primitiv rekursiven Funktion f gibt es $i \in \mathbb{N}$, so dass für all $x_1, \ldots, x_k \in \mathbb{N}^k$

$$f(x_1, \ldots, x_k) \leqslant B(i, \max\{x_1, \ldots, x_k\}).$$

Setze nun

$$A(x) = B(x, x).$$

Wie wir gerade gesehen haben, wird jede primitiv-rekursive Funktion von einem Zweig der Ackermannfunktion majorisiert. Hieraus folgt, dass A asymptotisch, d.h. für große x stärker als jede primitive-rekursive Funktion wächst.

Hieraus folgt, daß die Funktion A nicht primitiv rekursiv sein kann.

Hieraus folgt, daß die Funktion A nicht primitiv rekursiv sein kann.

Trotzdem ist die Ackermannfunktion intuitiv berechenbar (und total).

Die Ackermannfunktion genügt folgendem Rekursionsschema

$$B(0,x) = \begin{cases} 1, & \text{falls } x = 0 \\ 2, & \text{falls } x = 1 \\ x+2, & \text{sonst} \end{cases}$$

$$B(n+1,0) = 1$$

$$B(n+1,x+1) = B(n, B(n+1,x)).$$

Beim Schema der primitiven Rekursion wurde nur eine Variable als Rekursionsparameter benutzt, hier hingegen sind es zwei: n und x. Funktionen, die solchen Rekursionsgleichungen genügen, heißen *zweifach rekursiv*.

Um zu verdeutlichen, dass die Ackermannfunktion berechenbar ist, wollen wir beispielhaft die Berechnung von $B(2,2)$ betrachten. Dabei betrachten wir parallel eine kodierte Darstellung der Berechnung, in welcher wir uns merken, welche Zweige der Ackermannfunktion bzw. welche Argumente gerade „aktiv“ sind (man beachte die verdrehte Reihenfolge).

$B(2,2)$	kodiert:
$= B(2,2)$	$\rightarrow \langle 2,2 \rangle^*$
$= B(1,B(2,1))$	$\rightarrow \langle 1,2,1 \rangle^*$
$= B(1,B(1,B(2,0)))$	$\rightarrow \langle 0,2,1,1 \rangle^*$
$= B(1,B(1,1))$	$\rightarrow \langle 1,1,1 \rangle^*$
$= B(1,B(0,B(1,0)))$	$\rightarrow \langle 0,1,0,1 \rangle^*$
$= B(1,B(0,1))$	$\rightarrow \langle 1,0,1 \rangle^*$
$= B(1,2)$	$\rightarrow \langle 2,1 \rangle^*$
$= B(0,B(1,1))$	$\rightarrow \langle 1,1,0 \rangle^*$
$= B(0,B(0,B(1,0)))$	$\rightarrow \langle 0,1,0,0 \rangle^*$
$= B(0,B(0,1))$	$\rightarrow \langle 1,0,0 \rangle^*$
$= B(0,2)$	$\rightarrow \langle 2,0 \rangle^*$
$= 4$	$\rightarrow \langle 4 \rangle^*$

Bezeichnen wir die einzelnen kodierten Zwischenschritte der Rechnung als *Zustände*, dann gibt uns die nachfolgend definierte *Einzelschrittfunktion* $f : \mathbb{N} \to \mathbb{N}$ zu gegebenen Zustand $\langle x, n, \bar{u} \rangle^*$ mit $x, n \in \mathbb{N}$ und $\bar{u} \in \mathbb{N}^*$ den eindeutig bestimmeten Folgezustand an:

$$f(\langle x, 0, \bar{u} \rangle) = \langle B_0(x), \bar{u} \rangle^*$$
$$f(\langle 0, n+1, \bar{u} \rangle) = \langle 1, \bar{u} \rangle^*$$
$$f(\langle x+1, n+1, \bar{u} \rangle) = \langle x, n+1, n, \bar{u} \rangle^*$$

In allen anderen Fällen sei $f(v) = v$. Da die Funktion f durch Fallunterscheidung aus primitiv-rekursiven Funktionen hervorgeht, ist auch f selbst primitiv-rekursiv.

Für $x, n \in \mathbb{N}$ ist $f^t(\langle x, n \rangle^*)$ der t-te Zustand in der Berechnung von $B(n, x)$. Die Berechnung ist beendet, falls $\mathrm{lth}(f^t(\langle x, n \rangle^*) = 1$ gilt. Auf Grund unserer Ergebnisse über primitiv-rekursive Funktionen ist

$$g = \lambda n, x, t.f^t(\langle x, n \rangle^*)$$

primitiv-rekursiv. Für $B(n, x)$ suchen wir also das kleinste $t \in \mathbb{N}$ für das $\mathrm{lth}(g(n, x, t)) = 1$. Sei

$$h(n, x) = \text{das kleinste } t \in \mathbb{N} \text{ mit } \mathrm{lth}(g(n, x, t)) = 1$$

Man beachte, dass $B(n, x) = \pi_2(h(n, x))$. Im intuitiven Sinne ist h offensichtlich berechenbar.

Definition (Minimalisierung)

Für $f : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$, *sei* $\mu f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ *definiert durch*

$$\mu f(x_1, \ldots, x_k) = \begin{cases} \min \{y \mid f(x_1, \ldots, x_k, y) = 0 \\ \quad \wedge (\forall z < y) f(x_1, \ldots, x_k, z) \neq \bot\}, & \text{falls } \{\ldots\} \neq \varnothing \\ \bot & \text{sonst.} \end{cases}$$

Die Menge der μ-rekursiven Funktionen ist definiert wie die der primitiv rekursiven Funktion. Zusätzlich fordern wir aber auch

- Ist $f : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$ eine μ-rekursive Funktion, so auch μf.

Da alle primitiv rekursiven Funktionen total sind gibt es (partielle) μ-rekursive Funktionen, die nicht primitiv rekursiv sind. Wie wir mit der Ackermannfunktion schon gesehen haben gibt es sogar *totale* μ-rekursive Funktionen, die nicht primitiv rekursiv sind