(Master) Berechenbarkeit

v4.0.4.2.4.3 Ackermannfunktion ist nicht primitiv rekursiv

Kategory GmbH & Co. KG
Präsentiert von Jörg Kunze

Beschreibung

Inhalt. Die Ackermannfunktion ist nicht primitiv rekursiv. Wir haben aber schon gesehen, dass wir sie programmieren können. Das ergibt den Kern dieser Vorlesung: **der Begriff der primitiven Rekursion reicht nicht aus, um den Begriff der Berechenbarkeit zu formalisieren** . Und das ist der Grund, warum wir die μ-Rekursion eingeführt haben. Und es ist ein Grund, sich mit der Ackermannfunktion zu beschäftigen.

Für den Beweis zeigen wir, dass die Ackermannfunktion jede primitiv-rekursive Funktion majorisiert. Da eine Funktion sich nicht selbst majorisieren kann, ist das gewünschte gezeigt. Wir tun dies induktiv über die Definition von „primitiv rekursiv“.

Durch den Beweis bekommen wir auch ein Mittel, um aus dem Aufbau einer primitiv-rekursiven Funktionsdefinition den Parameter k zu errechnen, mit dem $A(k,\_)$ größer ist als die definierte Funktion.

Präsentiert. Von Jörg Kunze

Voraussetzungen. Primitiv rekursive Funktionen, Ackermannfunktion und ihre Eigenschaften.

Text. Der Begleittext als PDF und als LaTeX findet sich unter `https://github.com/kategory/kategoryMathematik/tree/main/v4%20Master/v4.0%20Berechenbarkeit/v4.0.4.2.4.3%20(Master)%20Berechenbarkeit%20-%20Ackermannfunktion%20ist%20nicht%20primitiv%20rekursiv`.

Meine Videos. Siehe auch in den folgenden Videos:
v4.0.4.2.4.2 (Master) Berechenbarkeit - Ackermannfunktion Eigenschaften
`https://youtu.be/MzDubMBlQ20`

v4.0.4.2.4.1 (Master) Berechenbarkeit - Ackermannfunktion ist mu-rekursiv
`https://youtu.be/cbZ8EJ4U-tg`

v4.0.4.2.4 (Master) Berechenbarkeit - Ackermannfunktionen
`https://youtu.be/XE-7YdHi2Vw`

v4.0.4.2 (Master) Berechenbarkeit - Rekursive Funktionen
`https://youtu.be/tFEn2GoLLEQ`

Quellen. Siehe auch in den folgenden Seiten:
`https://de.wikipedia.org/wiki/Ackermannfunktion`
`https://en.wikipedia.org/wiki/Ackermann_function`
`https://planetmath.org/ackermannfunctionisnotprimitiverecursive`
`https://planetmath.org/PropertiesOfAckermannFunction`

Buch. Grundlage ist folgendes Buch:
Computability
A Mathematical Sketchbook
Douglas S. Bridges
Springer-Verlag New York Inc. 2013
978-1-4612-6925-0 (ISBN)

1. Ackermannfunktion ist nicht primitiv rekursiv

1.1. **Eigenschaften.** Wir verwenden die in einem vorherigen Video gezeigten Definition und Eigenschaften er Ackermannfunktion (hierbei seien $k, n \in \mathbb{N}$):

(**D1**)	$A(0,n)$	$:= n+1$
(**D2**)	$A(k+1,0)$	$:= A(k,1)$
(**D3**)	$A(k+1,n+1)$	$:= A(k,A(k+1,n))$
(**E1**)	$A\colon \mathbb{N}^2 \to \mathbb{N}$	ist total
(**E2**)	$A(1,n)$	$= n+2$
(**E3**)	$A(2,n)$	$= 2n+3$
(**E4**)	$A(k,n)$	$> n$
(**E5**)	$A(k,n+1)$	$> A(k,n)$
(**E6**)	$A(k+1,n)$	$> A(k,n)$
(**E7**)	$A(k+1,n)$	$> A(k,n+1)$
(**E8**)	$A(k+k'+2,n)$	$> A(k,A(k',n))$
(**E9**)	$A(\max\{k,k'\}+4,n)$	$> A(k,n)+A(k,n)$

Aus den Eigenschaften **E5** und **E6** folgt die strenge Monotonie in beiden Parametern:

(**E5'**)	$n > m$	$\Rightarrow$	$A(k,n) > A(k,m)$
(**E6'**)	$l > k$	$\Rightarrow$	$A(l,n) > A(k,n)$

1.2. **Primitiv rekursiv.**

Definition 1.2.1. (Primitiv rekursiv): Wir definieren induktiv als **primitiv rekursiv**:

- (**PR1**): $f \circ g$, falls f, g primitiv rekursiv sind
- (**PR2**): h, mit $h(0,x_1,\cdots,x_s) := f(x_1,\cdots,x_s)$ und h, mit $h(n+1,x_1,\cdots,x_s) := g(n,h(n,x_1,\cdots,x_s),x_1,\cdots,x_s)$, falls f, g primitiv rekursiv sind
- (**PR3**): Die konstanten Funktionen $c(n) := c$ für alle $c \in \mathbb{N}$
- (**PR4**): Die Projektionen $P_j(x_1,\cdots,x_s) := x_j$
- (**PR5**): Die Nachfolger-Funktion $s(n) := n+1$.

1.3. **Die Ackermannfunktion ist nicht primitiv rekursiv.** Im Folgenden ist $\mathbf{x} := (x_1,\cdots,x_s)$ und $|\mathbf{x}| := \max\{x_1,\cdots,x_s\}$.

Definition 1.3.1. (majorisiert): Seien $g\colon \mathbb{N}^m \to \mathbb{N}$ und $h\colon \mathbb{N}^2 \to \mathbb{N}$, dann sagen wir h **majorisiert** g und wir schreiben $h > g$, falls

$$\exists k_g \forall \mathbf{x} |\mathbf{x}| > 1 \Rightarrow h(k_g, |\mathbf{x}|) > g(\mathbf{x}).$$

Hier betrachten wir h als Schar von Funktionen, parametrisiert über das erste Argument, und wir suchen das Element der Schar, das größer ist als die zu majorisierende Funktion.

Theorem 1.3.2. (keine Selbstmajorisierung): Sei $h\colon \mathbb{N}^2 \to \mathbb{N}$, dann gilt NICHT $h > h$.

Beweis. Siehe „v4.0.4.2.4.2 (Master) Berechenbarkeit - Ackermannfunktion Eigenschaften“. □

Theorem 1.3.3. (Ackermann majorisiert primitiv rekursiv): Sei $g\colon \mathbb{N}^s \to \mathbb{N}$ und A die Ackermannfunktion, dann gilt $A > g$.

Beweis. Siehe weiter unten. □

Theorem 1.3.4. (Ackermann nicht primitiv rekursiv): Die Ackermannfunktion A ist NICHT primitiv rekursiv.

Beweis. Wäre A primitiv rekursiv so müsste nach [Theorem 1.3.3, “Ackermann majorisiert primitiv rekursiv”] gelten $A > A$, was aber nach [Theorem 1.3.2, “keine Selbstmajorisierung”] nicht möglich ist. □

1.4. **Beweis von [Theorem 1.3.3, "Ackermann majorisiert primitiv rekursiv"].** Wir beweisen den Satz per vollständiger Induktion über den Aufbau der primitiv rekursiven Funktionen.

Theorem 1.4.1. (PR1: Verknüpfung): Sei $A(k_f, |\mathbf{x}|) > f(\mathbf{x})$ und $A(k_g, |\mathbf{x}|) > g(\mathbf{x})$. Setze $k_{f\circ g} := k_f + k_g + 2$. Dann gilt

$$A(k_{f\circ g}, |\mathbf{x}|) > f(g(\mathbf{x})).$$

Beweis.

$f(g(n))$	$<$	$A(k_f, g(\mathbf{x}))$	nach Definition von k_f
	$<$	$A(k_f, A(k_g, \|\mathbf{x}\|))$	nach E5' und Definition von k_g
	$<$	$A(k_f + k_g + 2, \|\mathbf{x}\|)$	nach E8

□

Theorem 1.4.2. (PR2: Rekursion): Sei $A(k_f, |\mathbf{x}|) > f(\mathbf{x})$ und $A(k_g, \max\{n, h(n, \mathbf{x}), |\mathbf{x}|\}) > g(n, h(n, \mathbf{x}), \mathbf{x})$. Dann gilt für die wie in PR2 rekursiv definierte Funktion h mit $k_{R(f,g)} := 5 + \max\{k_f, k_g\}$

$$A(k_{R(f,g)}, |\mathbf{x}|) > h(\mathbf{x}).$$

Beweis. Wir setzen $q := 1 + \max\{k_f, k_g\}$ und beweisen zunächst, als Zwischenbehauptung, über vollständige Induktion folgende Abschätzung:

(1) $$\boxed{h(n, \mathbf{x}) < A(q, n + |\mathbf{x}|)}.$$

Für $n = 0$ haben wir (unter Nutzung von E6')

$$h(0, \mathbf{x}) = f(\mathbf{x}) < A(k_f, |\mathbf{x}|) < A(q, 0 + |\mathbf{x}|).$$

Als Vorbereitung für den $n + 1$-Fall: Es gelten folgende beide Ungleichungen

$\max\{n, \|\mathbf{x}\|\}$	$<$	$n + \|\mathbf{x}\|$	$< A(q, n + \|\mathbf{x}\|)$	nach E4
$h(n, \mathbf{x})$			$< A(q, n + \|\mathbf{x}\|)$	nach Induktionsvoraussetzung

Diese beiden Ungleichungen ergeben zusammen

(2) $$\boxed{\max\{n, h(n, \mathbf{x}), |\mathbf{x}|\} < A(q, n + |\mathbf{x}|)}.$$

Für $n + 1$ können wir nun wie folgt abschätzen:

$h(n + 1, \mathbf{x})$	$=$	$g(n, h(n, \mathbf{x}), \mathbf{x})$	nach Definition von h
	$<$	$A(k_g, \max\{n, h(n, \mathbf{x}), \|\mathbf{x}\|\})$	nach Definition von k_g
	$<$	$A(k_g, A(q, n + \|\mathbf{x}\|))$	nach (2) und E5'
	$\leqslant$	$A(q - 1, A(q, n + \|\mathbf{x}\|))$	nach Definition von q und E6'
	$=$	$A(q, n + 1 + \|\mathbf{x}\|))$	nach D3 ,

welches die Zwischenbehauptung beweist. Damit können wir nun beweisen:

$h(n, \mathbf{x})$	$<$	$A(q, n + \|\mathbf{x}\|)$	nach Zwischenbehauptung (1)
	$<$	$A(q, 2 \cdot \max\{n, \|\mathbf{x}\|\} + 3)$	zunächst hätte +1 auch gereicht
	$<$	$A(k_g, A(2, \max\{n, \|\mathbf{x}\|\}))$	nach E3
	$<$	$A(q + 4, \max\{n, \|\mathbf{x}\|\})$	nach E8
	$<$	$A(5 + \max\{k_f, k_g\}, \max\{n, \|\mathbf{x}\|\})$	nach Definition von q

□

Theorem 1.4.3. (PR3: Konstante): Sei $c \in \mathbb{N}$ beliebig. Setze $k_c := c$. Dann gilt

$$A(k_c, n) > c.$$

Beweis. Sei $c := 0$. Dann gilt

$$A(0, n) = n + 1 > 0 \text{ (die Gleichheit nach D1).}$$

Sei nun $A(c, n) > c$. Dann gilt

$$A(c + 1, n) > A(c, n) > c \text{ (nach E6 und Voraussetzung).}$$

Da $A(c, n)$ mindestens 1 größer ist als c, muss $A(c + 1, n) > c + 1$ sein. □

Theorem 1.4.4. (PR4: Projektion): Setze $k_{P_j} := 0$. Dann gilt

$$A(k_{P_j}, |\mathbf{x}|) > P_j(\mathbf{x}).$$

Beweis. Sei $c := 0$. Dann gilt

$$A(k_{P_j}, |\mathbf{x}|) = A(0, |\mathbf{x}|) = |\mathbf{x}| + 1 > x_j = P_j(\mathbf{x}).$$

Die zweite Gleichheit wegen D1. □

Theorem 1.4.5. (PR5: Nachfolger): Setze $k_s := 1$. Dann gilt

$$A(k_s, n) > s(n)).$$

Beweis.

$$A(k_s, n) = A(1, n) = n + 2 > n + 1 = s(n).$$

Die zweite Gleichheit wegen E2. □

2. Schluss

Der Begriff der primitiv rekursiven Funktionen kann den Begriff der Berechenbarkeit nicht formalisieren. Die Ackermannfunktion ist berechenbar. Dies „wissen“ wir, da sie μ-rekursiv ist und alle μ-rekursiven Funktionen berechenbar sind. Wir wissen es auch, weil wir ein JavaScript-Programm zur Errechnung der Ackermannfunktion schreiben können, und dieses kann in eine Turing-Maschine überführt werden.

Dies ist ein Grund, sich mit der Ackermannfunktion zu beschäftigen. Ihr sehr schnelles Wachstum und damit das der der Funktion $A(n) := A(n, n)$ ist ein weiterer Grund.

Dies ist auch der Grund, warum wir in der Theorie der rekursiven Funktionen den Begriff der μ-Rekursion benötigen.

Wir können anhand der rekursiven Definition einer primitiv rekursiven Funktion das k für die Majorisierung durch $A(k, n)$ einfach berechnen. Dafür müssen wir nur die entsprechenden Formeln für das k in den Beweisschritten oben verwenden.

$k_{f \circ g}$	$:= k_f + k_g + 2$	(PR1)
$k_{R(f,g)}$	$:= 5 + \max\{k_f, k_g\}$	(PR2)
k_c	$:= c$	(PR3)
k_{P_j}	$:= 0$	(PR4)
k_s	$:= 1$	(PR5)

Wir wissen, dass die Ackermannfunktion für großes k abstrus schnell wächst. Mit dem hier gewählten Ansatz benötigen wir z.B. für die konstante Funktion $f(n) := 100$ eine extrem schnell wachsende Funktion, und ich habe den Eindruck, wir schießen hier mit Kanonen auf Spatzen.

Literatur

[Douglas2013] Douglas S. Bridges, *Computability, A Mathematical Sketchbook*, Springer, Berlin Heidelberg New York 2013, ISBN 978-1-4612-6925-0 (ISBN).

Symbolverzeichnis

$\mathbb{N}$	Die Menge der natürlichen Zahlen (mit Null): $\{0, 1, 2, 3, \cdots\}$
n, k, l, s, i, x_i, q	Natürliche Zahlen
$A(k, n)$	Ackermannfunktion
f, g, h	Funktionen
$s()$	Nachfolgerfunktion: $s(n) := n + 1$
$\mathbf{x}$	Vektor natürlicher Zahlen $(x_1, \cdots, x_s)$
$\lvert\mathbf{x}\rvert$	$\max\{x_1, \cdots, x_s\}$
$g > f$	Die Funktion g majorisiert die Funktion f
P_j	Die Projektionen $P_j(x_1, \cdots, x_s) := x_j$