Die Ackermannfunktion

Hans U. Simon (RUB)

Email: simon@lmi.rub.de

Homepage: http://www.ruhr-uni-bochum.de/lmi

Eine Frage zu Anfang

Ist jede intuitiv berechenbare totale (= total definierte) Funktion

$$f : \mathbb{N}^k \to \mathbb{N}$$

LOOP–berechenbar?

Im Jahre 1928 präsentierte Ackermann ein Gegenbeispiel.

Seine Idee: Entwerfe eine Funktion, die schneller wächst als jede LOOP-berechenbare Funktion.

Die Ackermannfunktion

Betrachte folgende induktiv definierte Funktion $a : \mathbb{N}^2 \rightarrow \mathbb{N}$:

$$
\begin{aligned}
a(0,y) &= y+1 \text{ für alle } y \geq 0 \\
a(x,0) &= a(x-1,1) \text{ für alle } x \geq 1 \\
a(x,y) &= a(x-1,a(x,y-1)) \text{ für alle } x,y \geq 1
\end{aligned}
$$

Die Definition besteht zwar nur aus drei Gleichungen, ist aber nicht ganz leicht zu durchschauen.

Ackermannfunktion (fortgesetzt)

Per „Salami–Taktik“ zerlegen wir $a(x, y)$ in „Scheiben“ $A_x : \mathbb{N} \rightarrow \mathbb{N}$:

$$A_x(y) := a(x, y)$$

und „schnuppern“ an $A_0, A_1, A_2, \ldots$.

Offensichtlich gilt $A_0(y) = y + 1$. Mit Induktion ergibt sich relativ leicht (s. Übung):

$$\begin{aligned}
A_1(y) &= y + 2 \\
A_2(y) &= 2y + 3 \\
A_3(y) &= 2^{y+3} - 3 \\
A_4(y) &= \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{y+3\ \text{Zweien}} - 3
\end{aligned}$$

Die Folge beginnt harmlos, nimmt aber dann schnell an Fahrt auf.

Macht die Definition von Ackermann Sinn ?

Wir beweisen durch „doppelte Induktion“, dass alle Funktionen A_x Werte aus $\mathbb{N}$ als Ergebnis liefern.

Induktionsanfang: $A_0(y) = y + 1 \in \mathbb{N}$ für alle $y \in \mathbb{N}$.

Induktionsvoraussetzung: $A_{x-1}(y) \in \mathbb{N}$ für alle $y \in \mathbb{N}$.

Induktionsschritt: Zum Studium von A_x erfolgt erneut eine vollständige Induktion (diesmal nach y):

Induktionsanfang: $A_x(0) = a(x, 0) = a(x - 1, 1) = A_{x-1}(1) \in \mathbb{N}$.

Induktionsvoraussetzung: $A_x(y - 1) \in \mathbb{N}$.

Induktionsschritt:

$$A_x(y) = a(x, y) = a(x - 1, a(x, y - 1)) = \overbrace{A_{x-1}(\underbrace{A_x(y - 1)}_{\in \mathbb{N}})}^{\in \mathbb{N}}$$

Folgerung: $a(x, y) = A_x(y) \in \mathbb{N}$ für alle $x, y \in \mathbb{N}$.

Ist die Ackermannfunktion berechenbar ?

Wir berechnen $a(3,0)$ mit Hilfe eines Kellerspeichers (Stapels):

										0	1				
				0	1				1	1	0	2			
		0	1	1	0	2		2	1	0	0	0	3		
0	1	2	1	0	0	0	3	1	0	0	0	0	0	4	
3	2	1	1	1	1	1	1	0	0	0	0	0	0	0	5

- Da wir mit „brain power“ $A_3(y) = 2^{y+3} - 3$ herausgefunden hatten, hätten wir das Ergebnis auch einfacher bestimmen können:

$$a(3,0) = A_3(0) = 2^{0+3} - 3 = 8 - 3 = 5 \ .$$

- Das Stapelverfahren (langsam wie es ist) funktioniert aber für alle möglichen Eingaben $(x,y) \in \mathbb{N}^2$.

Die allgemeine Vorgehensweise beim Stapelverfahren

Wir erinnern an die Definitionsgleichungen für die Ackermannfunktion:

$$\begin{aligned} a(0,y) &= y+1 \text{ für alle } y \geq 0 \\ a(x,0) &= a(x-1,1) \text{ für alle } x \geq 1 \\ a(x,y) &= a(x-1,a(x,y-1)) \text{ für alle } x,y \geq 1 \end{aligned}$$

Das Stapelverfahren interpretiert die zwei obersten Operanden x, y des Stapel (y oben) als $a(x,y)$ und manipuliert den Stapel wie folgt:

- Ersetze $0, y$ durch $y+1$ (Stapel wird niedriger).
- Für $x \geq 1$ ersetze $x, 0$ durch $x-1, 1$.
- Für $x, y \geq 1$ ersetze x, y durch $x-1, x, y-1$ (Stapel wird höher).

Dadurch wird $a(x,y)$ stets gemäß der Definitionsgleichungen ausgewertet.

Berechenbarkeit der Ackermannfunktion

Das Stapelverfahren ist leicht auf einer Mehrband–DTM implementierbar, die eines ihrer Bänder als Kellerspeicher verwendet.

Folgerung: Die Ackermannfunktion ist berechenbar.

In der Folge sagen wir einfach „berechenbar“ statt „Turing–berechenbar“ (oder „WHILE“– bzw. „GOTO–berechenbar“).

Monotonie–Eigenschaften der Ackermannfunktion

Die folgenden Ungleichungen gelten für alle $x, y \in \mathbb{N}$:

$$\begin{aligned} y &< a(x,y) && (1)\\ a(x,y) &< a(x,y+1) && (2)\\ a(x,y+1) &\leq a(x+1,y) && (3)\\ a(x,y) &< a(x+1,y) && (4) \end{aligned}$$

Insbesondere ist $a(x, y)$ streng monoton wachsend in x und y.

Beweis der 1. Ungleichung

Zum Beweis von $a(x, y) > y$ verwenden wir doppelte Induktion:

Induktionsanfang: $a(0, y) = y + 1 > y$ für alle $y \geq 0$.

Induktionsvoraussetzung: $a(x - 1, y) > y$ für alle $y \geq 0$.

Induktionsschritt: Analyse von $a(x, y)$ erfolgt mit Induktion nach y.

- **Induktionsanfang:** $a(x, 0) = a(x - 1, 1) > 1 > 0$.
- **Induktionsvoraussetzung:** $a(x, y - 1) > y - 1$ und somit $a(x, y - 1) \geq y$.
- **Induktionsschritt:** für alle $y \geq 0$ gilt dann

$$a(x, y) = a(x - 1, a(x, y - 1)) > a(x, y - 1) \geq y \ .$$

Beweis der 2. Ungleichung

Der Beweis von $a(x, y+1) > a(x, y)$ erfolgt durch eine Fallunterscheidung:

Fall 1: $x = 0$.

$$a(0, y+1) = y+2 > y+1 = a(0, y)$$

Fall 2: $x \geq 1$.

$$a(x, y+1) = a(x-1, a(x, y)) \stackrel{(1)}{>} a(x, y)$$

Beweis der 3. Ungleichung

Zum Beweis von $a(x+1,y) \geq a(x,y+1)$ verwenden wir Induktion nach y:

Induktionsanfang: $a(x+1,0) = a(x,1)$.

Induktionsvoraussetzung: $a(x+1,y-1) \geq a(x,y)$.

Induktionsschritt: Nun ergibt sich $a(x+1,y) \geq a(x,y+1)$ wie folgt:

$$
\begin{array}{rcll}
a(x+1,y) & = & a(x,\underbrace{a(x+1,y-1)}_{\overset{IV}{\geq} a(x,y)}) & \text{Definition der Ackermannfunktion} \\
& \geq & a(x,\underbrace{a(x,y)}_{\overset{(1)}{\geq} y+1}) & \text{Monotonie in } y \\
& \geq & a(x,y+1) & \text{Monotonie in } y
\end{array}
$$

Beweis der 4. Ungleichung

Die Ungleichung $a(x, y) < a(x + 1, y)$ ergibt sich direkt wie folgt:

$$a(x, y) \stackrel{(2)}{<} a(x, y + 1) \stackrel{(3)}{\leq} a(x + 1, y)$$

Wachstumsfunktion zu LOOP–Programmen

Betrachte ein LOOP–Programm P. P führt Anfangswerte $n_0, n_1, n_2, \ldots$ der Variablen in Endwerte über, die wir mit $n'_0, n'_1, n'_2, \ldots$ bezeichnen.

Folgende Funktion misst gewissermaßen, wie stark die Werte der Variablen durch Ausführung von P wachsen könnnen:

$$f_P(n) := \max \left\{ \sum_{i \geq 0} n'_i \;\middle|\; \sum_{i \geq 0} n_i \leq n \right\}$$

In Worten: $f_P(n)$ ist die größtmögliche Summe der Variablenendwerte, wenn die Summe der Variablenanfangswerte durch n beschränkt ist.

Wachstumsfunktion (fortgesetzt)

Wenn P eine Funktion $g : \mathbb{N} \to \mathbb{N}$ berechnet, dann gilt offensichtlich

$$g(n) \leq f_P(n) \ ,$$

da die Anfangskonfiguration zu Eingabe n mit einer speziellen Wahl der Variablenanfangswerte, nämlich

$$n_0 = 0, n_1 = n \text{ und } n_i = 0 \text{ für alle } i \geq 2 \ ,$$

operiert, welche bei dem Maximum in der Definition von f_P berücksichtigt wird.

Verhältnis von Ackermannfunktion und LOOP–Programmen

Schlüsselresultat: Für jedes LOOP–Programm P gilt:

$$\exists k \geq 0, \forall n \geq 0 : f_P(n) < A_k(n)$$

Die einem LOOP–Programm P zugeordnete Funktion f_P wächst demnach nicht schneller als die k–te „Schicht" $A_k(\cdot)$ der Funktion $a(\cdot,\cdot)$.

Folgerung: Die Ackermannfunktion ist nicht LOOP–berechenbar.

Widerspruchsbeweis zur Folgerung

(heuchlerische) Annahme: $a(x, y)$ ist LOOP–berechenbar.

- Dann ist auch $g(n) := a(n, n)$ LOOP–berechenbar, sagen wir mit LOOP–Programm P.
- Es gilt dann $g(n) \leq f_P(n)$ für alle $n \geq 0$.
- Wähle dann k so aus, dass $f_P(n) < A_k(n)$ für alle $n \geq 0$.
- Für $n = k$ ergibt sich nun ein Widerspruch:

$$g(k) \leq f_P(k) < A_k(k) = a(k, k) = g(k)$$

Beweis des Schlüsselresultates

Der Beweis ist aufgebaut wie folgt:

- Reduktion des Schlüsselresultates auf sogenannte „einfache“ LOOP–Programme.
- Beweis des Schlüsselresultates für einfache LOOP–Programme mit Hilfe von struktureller Induktion.

Einfache LOOP–Programme

Ein LOOP–Programm heißt einfach **gdw** es die folgenden Bedingungen erfüllt:

- Anweisungen der Form „$x_i := x_j \pm c$“ verwenden nur Konstanten $c \in \{0, 1\}$.
- Bei Anweisungen der Form „LOOP x_i DO Q END“ wird Variable x_i nicht innerhalb von Q verwendet.

Jedes LOOP–Programm ist in ein äquivalents einfaches LOOP–Programm transformierbar:

- Eine Anweisung „$x_i := x_j \pm c$“ mit $c \geq 2$ kann durch c–fache Hintereinanderausführung von „$x_i := x_j \pm 1$“ simuliert werden.
- Eine Anweisung der Form „LOOP x_i DO Q END“, die x_i innerhalb Q verwendet, kann statt x_i eine bisher unbenutzte Variable y verwenden, wobei der LOOP–Anweisung mit Laufvariable y die Wertzuweisung $y := x_i + 0$ vorangeschaltet wird.

Induktionsanfang

P hat die Form „$x_i := x_j \pm c$“ mit $c \in \{0,1\}$.

Das größte Wachstum wird erzielt für Variablenanfangswerte

$$n_i = 0, n_j = n$$

und die Anweisung „$x_i := x_j + 1$“, wobei sich Variablenendwerte

$$n_i' = n + 1, n_j = n$$

ergeben (restliche Variablen auf Null). Somit gilt:

$$f_P(n) \leq 2n + 1 < 2n + 3 = A_2(n) \ .$$

1. Induktionsschritt

P hat die Form „$P_1; P_2$“.

Induktionsvoraussetzung: Es gibt Konstanten k_1, k_2 so dass

$$f_{P_1}(n) < A_{k_1}(n) \text{ und } f_{P_2}(n) < A_{k_2}(n)$$

für alle $n \geq 0$.

Offensichtlich gilt

$$f_P(n) \overset{IV}{\leq} f_{P_2}(f_{P_1}(n)) < A_{k_2}(f_{P_1}(n)) < A_{k_2}(A_{k_1}(n)) = a(k_2, a(k_1, n)) \ .$$

Fall 1: $k_1 \leq k_2 + 1$.

$$a(k_2, a(k_1, n)) \leq a(k_2, a(k_2+1, n)) = a(k_2+1, n+1) \leq a(k_2+2, n) = A_{k_2+2}(n)$$

Fall 2: $k_1 > k_2 + 1$.

$$a(k_2, a(k_1, n)) < a(k_1 - 1, a(k_1, n)) = a(k_1, n+1) \leq a(k_1+1, n) = A_{k_1+1}(n)$$

Für $k := \max\{k_1 + 1, k_2 + 2\}$ gilt dann $f_P(n) < A_k(n)$ für alle $n \geq 0$.

2. Induktionsschritt

P hat die Form „LOOP x_i DO Q END“, wobei x_i in Q nicht verwendet wird.

Induktionsvoraussetzung: Es gibt eine Konstante k so dass $f_Q(n) < A_k(n)$ für alle $n \geq 0$.

Wähle Variablenanfangswerte $n_0, n_1, n_2, \ldots$ mit $f_P(n) = \sum_{j \geq 0} n_i'$ und setze $m := n_i$ (Anfangswert von x_i, der zu maximaler Summe der Variablenendwerte führt).

Dann gilt

$$f_P(n) \leq f_Q^m(n-m) + m \ .$$

Erkläre !

Ausgehend von dieser Ungleichung und $f_Q(n) < A_k(n)$ werden wir

$$f_P(n) < A_{k+1}(n) \ , \tag{5}$$

nachweisen, was den Induktionsbeweis abschließt.

„High Noon“ bei Familie Ackermann

$$
\begin{aligned}
f_P(n) &\le f_Q^m(n-m)+m \\
&\le A_k(f_Q^{m-1}(n-m)+(m-1) \\
&\le A_k^2(f_Q^{m-2}(n-m))+(m-2) \\
&\dots \\
&\le A_k^m(n-m) \\
&= A_k^{m-2}\,(a(k,a(k,n-m))) \\
&< A_k^{m-2}(\underbrace{a(k,a(k+1,n-m))}_{=a(k+1,n-m+1)}) \\
&= A_k^{m-3}(\underbrace{a(k,(a(k+1,n-m+1)))}_{=a(k+1,n-m+2)}) \\
&\dots \\
&= a(k+1,n-1) \\
&< a(k+1,n) = A_{k+1}(n)
\end{aligned}
$$