Gliederung

1. Einführung
2. Berechenbarkeitsbegriff
3. LOOP-, WHILE-, und GOTO-Berechenbarkeit
4. Primitive und partielle Rekursion
5. Grenzen der LOOP-Berechenbarkeit
6. (Un-)Entscheidbarkeit, Halteproblem und Reduzierbarkeit
7. Das Postsche Korrespondenzproblem
8. Komplexität – Einführung
9. NP-Vollständigkeit
10. PSPACE

Grenzen der LOOP-Berechenbarkeit

Wissen: alle LOOP-berechenbaren Funktionen sind total

Frage: gibt es totale Funktionen die nicht LOOP-berechenbar sind? Ja! (Diagonalisierung)

Theorem

Sei L eine Liste aller 1-stelligen, LOOP-berechenbaren Funktionen. Dann ist $g : \mathbb{N} \to \mathbb{N}$ mit

$$g(n) := L_n(n) + 1$$

total und nicht LOOP-berechenbar.

Beweis

Wäre g LOOP-berechenbar, so gäbe es ein $k \in \mathbb{N}$ mit $L_k = g$.

$\leadsto g(k) = L_k(k) + 1 = g(k) + 1$

Aber: Ist g (Turing-)berechenbar?

Ackermannfunktion I

Die Ackermannfunktion (Variante Rósza Péter):

$$
\begin{aligned}
\mathrm{ack}(0, y) &:= y + 1, \\
\mathrm{ack}(x, 0) &:= \mathrm{ack}(x-1, 1), \\
\mathrm{ack}(x, y) &:= \mathrm{ack}(x-1, \mathrm{ack}(x, y-1)) \\
&= \underbrace{\mathrm{ack}(x-1, \mathrm{ack}(x-1, \mathrm{ack}(x-1, \dots, \mathrm{ack}(x-1, 1))) \dots)}_{(y+1)\text{ mal}}
\end{aligned}
$$

Die Ackermannfunktion wächst extrem schnell (z.B. gilt $\mathrm{ack}(4, 2) \approx 2 \cdot 10^{19728}$).

Eine „modernisierte“ Variante:

$$
\begin{aligned}
a(0, y) &:= 1, \\
a(1, y) &:= 3y + 1, \\
a(x, y) &:= \underbrace{a(x-1, a(x-1, \dots, a(x-1, y) \dots)}_{y\text{ mal}}
\end{aligned}
$$

Beobachtung: a ist total und in beiden Argumenten monoton wachsend

Frage: können Sie zeigen, dass $a(x, y) \leq \mathrm{ack}(x, 3y)$? **(*)**

Ackermannfunktion II

Theorem

Die Ackermannfunktion a ist nicht LOOP-berechenbar.

Strategie: zeigen, dass a schneller wächst als jede LOOP-berechenbare Funktion.
$\rightsquigarrow$ definieren zunächst $f_P(n)$ als die maximale Summe aller Variablenendwerte, die das Programm P erzeugen kann, wenn die initiale Belegung höchstens Summe n hat.

Definition

Sei P ein LOOP-Programm, welches die Variablen $x_0, x_1, \ldots, x_k$ verwendet.
Die i**'te Speicherüberführungsfunktion** $F_i^P : \mathbb{N}^{k+1} \to \mathbb{N}$ an der Stelle $(n_0, n_1, \ldots, n_k)$ ist der Wert von x_i am Ende des Programmes P falls P mit $x_j = n_j$ für alle $0 \leq j \leq k$ gestartet wird.
Außerdem sei die Funktion $f_P : \mathbb{N} \to \mathbb{N}$ definiert als

$$f_P(n) := \max \left\{ \sum_{i=0}^{k} F_i^P(n_0, \ldots, n_k) \;\middle|\; \sum_{i=0}^{k} n_i \leq n \right\}.$$

Ackermannfunktion III

Lemma

Zu jedem LOOP-Programm P existiert ein $\ell \in \mathbb{N}$ sodass für alle $n \geq \ell$ gilt: $f_P(n) < a(\ell, n)$.

Beweis (Induktion über Termstruktur von P)

Fall 1: $P =$ „$x_i := x_j \pm c$“
$\rightsquigarrow f_P(n) \leq 2n + c \leq 3n < 3n + 1 = a(1, n)$.$\rightsquigarrow$ Wähle $\ell := \max\{c, 1\}$.

Fall 2: $P =$ „$P_1; P_2$“
Induktionsvoraussetzung $\rightsquigarrow$ es gibt $\ell_1, \ell_2 \in \mathbb{N}$, sodass für alle $n \geq \max\{\ell_1, \ell_2\} =: \ell_3$ gilt:
$f_{P_1}(n) < a(\ell_1, n) \leq a(\ell_3, n)$ und $f_{P_2}(n) < a(\ell_2, n) \leq a(\ell_3, n)$.

Da $f_P(n) \leq f_{P_2}(f_{P_1}(n))$ folgt (falls $\ell_3 \geq 2$):
$$f_P(n) < a(\ell_3, f_{P_1}(n)) \leq a(\ell_3, a(\ell_3, n)) \leq \underbrace{a(\ell_3, a(\ell_3, \ldots, a(\ell_3, n) \ldots)}_{n-\text{mal}} = a(\ell_3 + 1, n).$$

$\rightsquigarrow$ Wähle $\ell := \max\{\ell_3 + 1, 2\}$.

Ackermannfunktion III

Lemma

Zu jedem LOOP-Programm P existiert ein $\ell \in \mathbb{N}$ sodass für alle $n \geq \ell$ gilt: $f_P(n) < a(\ell, n)$.

Beweis (Induktion über Termstruktur von P)

Fall 3: $P =$ „**LOOP** x_i **DO** P' **END**“

Induktionsvoraussetzung $\leadsto$ es gibt $\ell' \in \mathbb{N}$, sodass für alle $n \geq \ell'$ gilt: $f_{P'}(n) < a(\ell', n)$.

Dann gilt $f_P(n) \leq (\underbrace{f_{P'} \circ \ldots \circ f_{P'}}_{n \text{ mal}})(n) < \underbrace{a(\ell', a(\ell', \ldots, a(\ell', n) \ldots)}_{n \text{ mal}} = a(\ell' + 1, n)$.

$\leadsto$ Wähle $\ell := \ell' + 1$.

Ackermannfunktion IV

Lemma

Zu jedem LOOP-Programm P existiert ein $\ell \in \mathbb{N}$ sodass für alle $n \geq \ell$ gilt: $f_P(n) < a(\ell, n)$.

Theorem

Die Ackermannfunktion a ist nicht LOOP-berechenbar.

Beweis

Annahme: a LOOP-berechenbar.
$\rightsquigarrow g(n) := a(n, n)$ LOOP-berechenbar vermöge LOOP-Programm P.
$\rightsquigarrow$ es gibt ein $\ell \in \mathbb{N}$, sodass für alle $n \geq \ell$ gilt: $f_P(n) < a(\ell, n)$.
$\rightsquigarrow g(\ell) \leq f_P(\ell) < a(\ell, \ell) = g(\ell)$.

Bemerkung: Funktion g im Beweis wächst schneller als jede LOOP-berechenbare Funktion

Ein WHILE-Programm für die Ackermannfunktion

Theorem

a ist WHILE-berechenbar.

$$
\begin{aligned}
a(0, y) &:= 1,\\
a(1, y) &:= 3y + 1,\\
a(x, y) &:= \underbrace{a(x-1, a(x-1, \ldots, a(x-1, y)\ldots)}_{y \text{ mal}}
\end{aligned}
$$

Beweis

Idee: Rekursion in WHILE-Schleife pressen

Schwierigkeit: Unbeschränkte Rekursionstiefe mit beschränkter Anzahl Variablen!

$\rightsquigarrow$ speichern mehrere Zahlen in einer Variable.

$\rightsquigarrow$ injektive, LOOP-berechenbare „Pairing-Funktion“, z.B. $c(x, y) := 2^{x+y} + x$

(Umkehrfunktionen $\text{first}(c(x, y)) := x$ und $\text{second}(c(x, y)) := y$ LOOP-berechenbar)

$\rightsquigarrow$ Kellerinhalt $n_1, n_2, \ldots, n_k$ in Zahl $n := c(n_1, c(n_2, \ldots, c(n_k, 0)\ldots))$ gespeichert

$$
\begin{aligned}
\text{INIT} &\rightsquigarrow n := 0\\
\text{PUSH}(x) &\rightsquigarrow n := c(x, n)\\
\text{POP} &\rightsquigarrow x := \text{first}(n);\, n := \text{second}(n);\, \textbf{return } x
\end{aligned}
$$

Ein WHILE-Programm für die Ackermannfunktion

Theorem

a ist WHILE-berechenbar.

$$a(0,y) := 1,$$
$$a(1,y) := 3y+1,$$
$$a(x,y) := \underbrace{a(x-1, a(x-1, \ldots, a(x-1, y)\ldots)}_{y \text{ mal}}$$

Beweis

```
INIT; PUSH(x); PUSH(y);
while STACK SIZE > 1 do // second(n) ≠ 0
    y ← POP;
    x ← POP;
    if x = 0 then
        PUSH(1)
    else if x = 1 then
        PUSH(3 · y + 1)
    else
        LOOP y DO PUSH(x − 1) END;
        PUSH(y)
x0 ← POP;
```