

BERECHENBARKEIT UND KOMPLEXITÄT
UNIVERSITÄT LEIPZIG
SOMMERSEMESTER 2008

ADRIAN IMMANUEL KIESS

Vorlesung vom 7. April 2008.

Teil 1. Berechenbarkeitsbegriffe

1. LOOP-BERECHENBARKEIT

1.1. LOOP-Programme.

1.1.1. *Syntax.*

1.1.2. *Semantik.* Eine Funktion $f : \mathbb{N}^l \rightarrow \mathbb{N}$ heißt LOOP-Berechenbar, wenn es ein LOOP-Programm P mit folgenden Eigenschaften für alle $n_1, \dots, n_l \in \mathbb{N}$:

Belegt man die Variable x_i mit $\begin{cases} n_i & \text{für } 1 \leq i \leq l \\ 0 & i > l \end{cases}$

und führt P aus,

so ist $f(n_1, \dots, n_l)$ genau der Wert von x_1 .

Addition.

$j > 1 \quad x_1 := x_i + 0$
 $LOOP x_j DO x_1 := x_1 + 1 END$
berechnet die Fkt. $(n_1, \dots, n_l) \mapsto n_i + n_j$

Multiplikation.

$x_1 := 0$
 $LOOP x_j DO \underbrace{x_1 := +x_i}_{\text{Abk. für obig. LOOP-Programm}} END$
berechnet die Fkt. $(n_1, \dots, n_l) \mapsto n_i \times n_j$

Division.

$x_i := 0 \quad x_i := x_i + 1;$
 $LOOP x_i DO x_1 x_i - x_j; x_1 := x_1 + 1;$
 $IF X_i = 0 THEN x_i := x_1 - 1 END$
berechnet die Fkt $(n_i, \dots, n_l) \mapsto n_i DIV n_j$

If-Else.

$y := 1; x' := x$
 $LOOP y DO P END$
 $\# \quad y := 0$
 $LOOP x' DO : y := 1 END$
 $LOOP y DO P' END$
beschränkt $IF x = 0 THEN P ELSE P' END$

Vorlesung vom 14.04.2008.

2.2 Die Ackermann-Funktion. Die folgenden Funktionen sind LOOP-berechenbar.

$$(m, n) \mapsto m + n$$

$$(m, n) \mapsto m \times n$$

$$(m, n) \mapsto m^n$$

$$(m, n) \mapsto m^{n^n}$$

\implies LOOP-berechenbare Funktionen können sehr schnell wachsen, die Ackermann-Funktion wächst noch schneller.

Konstruktion: Wir definieren eine Folge von Funktionen $a_x \times \mathbb{N} (x \in \mathbb{N})$ induktiv durch

- $a_0(y) = y + 1 \quad (y \in \mathbb{N})$
- $a_x(y) \begin{cases} a_{x-1}(1) & \text{falls } y = 0 \\ a_{x-1}(a_x(y-1)) & \text{falls } y \geq 1 \end{cases} \quad (y \in \mathbb{N}, x \geq 1)$

Lemma. 2.3:

- (1) Ist $x > 0$ und $x \in \mathbb{N}$, so gilt $a_x(y) = a_{x-1}(1)$.
- (2) Für alle $x \in \mathbb{N}$ ist a_x LOOP-berechenbar.

Definition. 2.4.

Die Funktion $a : \mathbb{N} \mapsto \mathbb{N}$ heißt *Ackermann-Funktion*.

$$(x, y) \mapsto a_x(y)$$

(1928, Fassung von Hermes)

Ackermann-Ziel ist intuitiv berechenbar.

Betrachte das folgende Stack-Programm:

```

stack = ε;
push(x); push(y)
WHILE size(STACK) > 1 DO
  pop(y); pop(x)
  if x=0 then push(y+1)
    else if y=0 then push(x-1); push(1);
      else push(x-1); push(y-1); end
  end
ENDWHILE
pop(x)
output(x)

```

Schreibweise: $n_1, n_2 \dots n_l \rightsquigarrow m_1, m_2 \dots m_l$

falls für alle $n_{l+1}, n_{l+2}, \dots, n_k$ gilt:

Wenn am Eingang der Schleife $\text{stack} = n_1, n_2, \dots, n_k$ gilt, so wird zu einem späteren Zeitpunkt die Schleife betreten mit $\text{stack} = m_1, m_2, \dots, m_n, n_{l+1} \dots, n_k$

Lemma. 2.5.

Für alle $x, y \in \mathbb{N}$ gilt $y, x \rightsquigarrow a_x(y)$

Vorlesung vom 21.4.2008.

Definition. 2.8. P LOOP-Programm

Lemma. 2.8. P LOOP-Programm

Fakt. Satz 2.9. Die Ackermann-Funktion ist nicht LOOP-berechenbar.

Beweis. Indirekt angenommen, Q wäre ein LOOP-Programm, das a berechnet.

Sei $P = (x_2 := x_1; Q)$

$\implies P$ ist ein LOOP-Programm, das die Fkt.

$g : \mathbb{N} \rightarrow \mathbb{N}$ berechnet.

$$n \mapsto a(n, n)$$

$$\implies \forall n \in \mathbb{N} : f_p(k) \geq a(n, n)$$

□

Lemma. 2.8. $\Rightarrow \exists k \in \mathbb{N} \quad \forall n \in \mathbb{N} : f_p(a) < a_k(k)$

Wähle $n \in k$

$\Rightarrow a(k, k) \leq f_p(k) < a_k(k) = a(k, k) \Rightarrow$ WIEDERSPRUCH. (Induktion)

WHILE UND GOTO-BERECHENBARKEIT

WHILE-Berechenbarkeit.

Syntaxdefinition der WHILE-Programme.

- (1) Sind $i, j \geq 1$ und $c \in \{0, 1\}$, so sind
 $x_i := x, x_i := x_j + c, x_i := x_j - c$ WHILE-Programme
- (2) Sind P_1, P_2 WHILE-Programme, so auch $P_i; P_2$
- (3) Ist $i \in \mathbb{N}$ und P ein WHILE-Programm, so sind auch $LOOP x_i DO P END$
 und $WHILE x_i + 0 DO END$ WHILE-Programme.

Partielle Funktionen. Seien A, B Mengen, $T \subseteq A$ und $f : T \rightarrow B$ eine Funktion (Abb.). Dann heißt f auch **partielle Funktion von A und B** .

Bez. $f : A \dashrightarrow B \quad Def(f) = T$

Beispiel. 3.2. Das WHILE-Programm $WHILE x_1 \neq 0 DO x_1 := 1 END$ berechnet die partielle Funktion $f : \mathbb{N} \dashrightarrow \mathbb{N}, n \mapsto \begin{cases} 0 & n = 0 \\ def & n \geq 1 \end{cases}$

LOOP-Programm ist auch WHILE-Programm.

GOTO-Berechenbarkeit.

Syntaxdefinition der GOTO-Programme.

Anweisungen::

- (1) HALT ist eine Anweisung
- (2) Sind $i, j \geq 1$ und $c \in \{0, 1\}$, so sind
 $x_i := c, x_i := x_j + c, x_i := x_j - c$ Anweisungen
- (3) Sind $i, j \geq 1$ und $c \in \mathbb{N}$, so sind $GOTO i$ und $IF x_i = c THEN GOTO j$
 Anweisungen

Vorlesung vom 26.4.2008.

Semantik der GOTO-Programme. Sei P ein GOTO-Programm mit Variablen x_1, \dots, x_k und $n+1$ Anweisungen. Eine Konfiguration ist ein Tupel $(z, y_1, \dots, y_k) \in \mathbb{N}^{k+1}$ mit $1 \leq z \leq n+1$. Für zwei Konfigurationen $c = (z, y_1, \dots, y_k), C' = (z', y'_1, \dots, y'_k)$ gelte $C \models C'$, falls:

- $A_z = HALT$ und $C = C'$.
- $A_z = (x_i := x), z' = z + 1, y'_i = x$ und $y'_l = y_l$ für alle $l + i$.
- $A_z = (x_i := x_j + c), z' = z + 1, y'_i = x_i + c$ und $y'_l = y_l$ für alle $l + i$.
- $A_z = (x_i := x_j - c), z' = z + 1, y'_i = \max\{0, y_i - c\}, y'_l = y_l$ für alle $l + i$.
- $A_z = GOTO i, z' = i, y'_l = y_l$ für alle $1 \leq l \leq k$.
- $A_z = IF x_i = c THEN GOTO j$
 - $y_i = x, z' = j, y'_l = y_l$ für alle $1 \leq l \leq k$.
 - $y_i \neq c, z' = z + 1, y'_l = y_l$ für alle $1 \leq l \leq k$.

Übersetzung von GOTO-Anweisungen A in WHILE-Programme A':

A	A'
$x_i := c$	$(A; z := z + 1)$
$x_i = x_j \pm c$	
$GOTO i$	$z := i$
$IF x_i = 0 THEN GOTO j$	$IF x_i = 0 THEN z := j ELSE z := z + 1 END$; dies ist ein LOOP-Programm!
$HALT$	$z := 0$

Vorlesung vom 16.06.2008.

$$w \in \{0, 1\}^* \curvearrowright M_w = \begin{cases} M & \text{falls } w \text{ Kodierung einer 1-Band-TM } M \text{ ist} \\ \bar{M} & \text{falls } w \text{ keine Kodierung einer TM ist} \end{cases}$$

Definition. 6.1. $k = \{w \in \{0, 1\}^* \mid M_w \text{ hält bei Eingabe von } w\}$

Bemerkung. 6.2. k ist nicht entscheidbar.

Beweis. Indirekt Ang., χ_K wird von 1-Band-TM M berechnet.

1-Band-TM

M' stoppt bei Eingabe von $w \Leftrightarrow M$, gibt 0 aus bei Eingabe von w .

Sei w' Kodierung von M' .

$$M' = M_{w'}$$

$\Leftrightarrow M$ gibt 0 bei Eingabe von w' aus.

$\Leftrightarrow \chi_K(w') = 0$

$\Leftrightarrow w' \notin K$

$\Leftrightarrow M_{w'}$ hält nicht bei Eingabe von w'

$\Leftrightarrow M'$ stoppt bei Eingabe von w' nicht

\Leftrightarrow Beh. □

Klar: K ist semi-entscheidbar.

Bemerkung. 5.5. $\Rightarrow \Sigma^* \setminus K = \{w \in \{0, 1\}^* \mid M_w \text{ hält bei Eingabe von } w \text{ nicht}\}$ ist nicht semi-entscheidbar.

Definition. 6.3. Seien $A, B \in \Sigma^*$.

A heißt auf B reduzierbar.

$A \leq B$

" B ist mindestens so schwer wie A "

$\Leftrightarrow \exists$ berechenbare totale Fkt. $f: \Sigma^* \rightarrow \Sigma^*$ mit:

– $\forall w \in \Sigma^* : w \in A \Leftrightarrow f(w) \in B$.

Beispiel. 6.4. Sei $H := \{w\#x \mid M_w \text{ hält bei Eingabe von } x\}$

= das (allgemeine) Halteproblem

Denn die Fkt. $f: \Sigma^* \rightarrow \Sigma^*$ ist berechenbar und:

$$w \mapsto w\#w$$

$\forall_w \in \dots$ (abgewischt); -)

Lemma. 6.7. (Satz von Rice). Sei $\mathfrak{F} \neq \emptyset$ eine echte Teilmenge der Menge der berechenbaren Funktionen.

$\Rightarrow C(\mathfrak{F}) = \{w \in \Sigma^* \mid M_w \text{ ist deterministisch und berechnet eine Funktion aus } \mathfrak{F}\}$ ist nicht entscheidbar.

Beweis. Sei Ω die leere (nirgends definierte) Funktion

Fall 1. 1. $\Omega \in \mathfrak{F}$

\mathfrak{F} die echte Teilm. $\Rightarrow \exists$ det TN Q , die eine Fkt. $q \notin \mathfrak{F}$ berechnet.

Für $w \in \Sigma^*$ sei

$M^W : \text{START} \rightarrow M_w(2, 2) \rightarrow Q(1, 2) \rightarrow \text{STOP}$

M^w berechnet Fkt. g^w . Denn:

...

□

E-mail address: adrian@immanuelK.net

URL: <http://www.immanuelK.net>