Literatur zur Theoretischen Informatik

U. Schöning: Theoretische Informatik kurz gefaßt. B.I.-Wissenschaftsverlag, 1992.

K. Wagner: Einführung in die Theoretische Informatik. Springer-Verlag, 2003.

I. Wegener: Theoretische Informatik. Teubner-Verlag, 1993.

J. E. Hopcroft, J. D. Ullman: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie. 2. Aufl., Addison-Wesley, 1990.

G. Vossen, K.-U. Witt: Grundlagen der Theoretischen Informatik mit Anwendungen. Vieweg-Verlag, Braunschweig, 2000.

A. Asteroth, Ch. Baier: Theoretische Informatik. Pearson Studium, 2002.

Intuitiver Algorithmenbegriff

Ein Algorithmus

- überführt Eingabedaten in Ausgabedaten (wobei die Art der Daten vom Problem, das durch den Algorithmus gelöst werden soll, abhängig ist),

- besteht aus einer Folge von Anweisungen mit folgenden Eigenschaften:
  - es gibt eine eindeutig festgelegte Anweisung, die als erste auszuführen ist,
  - nach Abarbeitung einer Anweisung gibt es eine eindeutig festgelegte Anweisung, die als nächste abzuarbeiten ist, oder die Abarbeitung des Algorithmus ist beendet und hat eindeutig bestimmte Ausgabedaten geliefert,
  - die Abarbeitung einer Anweisung erfordert keine Intelligenz (ist also prinzipiell durch eine Maschine realisierbar).

LOOP/WHILE–Programme – Definition

Grundsymbole: 0, S, P, **LOOP**, **WHILE**, **BEGIN**, **END**, $:=$, $\neq$, ;, (,)

Variablensymbole: $x_1, x_2, \ldots, x_n, \ldots$

Definition:

i) Eine Wertzuweisung ist ein Wort, das eine der folgenden vier Formen hat:

$x_i := 0$ für $i \in \mathbf{N}$, $x_i := S(x_j)$ für $i \in \mathbf{N}, j \in \mathbf{N}$,

$x_i := x_j$ für $i \in \mathbf{N}, j \in \mathbf{N}$, $x_i := P(x_j)$ für $i \in \mathbf{N}, j \in \mathbf{N}$

Jede Wertzuweisung ist ein Programm.

ii) Sind Π, Π_1 und Π_2 Programme und x_i eine Variable, $i \in \mathbf{N}$, so sind auch die folgenden Wörter Programme:

$\Pi_1; \Pi_2$, **LOOP** x_i **BEGIN** Π **END** , **WHILE** $x_i \neq 0$ **BEGIN** Π **END**

.

LOOP/WHILE–Programme – Beispiele

a) **LOOP** x_2 **BEGIN** $x_1 := S(x_1)$ **END** ,

b) $x_3 := 0$;
LOOP x_1 **BEGIN**
LOOP x_2 **BEGIN** $x_3 := S(x_3)$ **END**
END

c) **WHILE** $x_1 \neq 0$ **BEGIN** $x_1 := x_1$ **END** ,

d) $x_3 := 0$; $x_3 := S(x_3)$;
WHILE $x_2 \neq 0$ **BEGIN**
$x_1 := 0$; $x_1 := S(x_1)$; $x_2 := 0$; $x_3 := 0$
END ;
WHILE $x_3 \neq 0$ **BEGIN** $x_1 := 0$; $x_3 := 0$ **END**.

LOOP/WHILE – Berechenbarkeit

Definition:
Π sei ein Programm mit n Variablen. Für $1 \leq i \leq n$ bezeichnen wir mit $\Phi_{\Pi,i}(a_1, a_2, \ldots, a_n)$ den Wert, den die Variable x_i nach Abarbeitung des Programms Π annimmt, wobei die Variable x_j, $1 \leq j \leq n$, als Anfangsbelegung den Wert a_j annimmt.

Π werden dadurch n Funktionen $\Phi_{\Pi,i}(x_1, x_2, \ldots, x_n)$, $1 \leq i \leq n$, zugeordnet

Definition:
Eine Funktion $f(x_1, x_2, \ldots, x_n)$ heißt **LOOP/WHILE**-berechenbar, wenn es ein Programm Π mit m Variablen, $m \geq n$, derart gibt, dass

$$\Phi_{\Pi,1}(x_1, x_2, \ldots, x_n, 0, 0, \ldots, 0) = f(x_1, x_2, \ldots, x_n)$$

gilt.

Berechenbarkeit der Fibonacci-Funktion

Fibonacci-Funktion: $f(0) = f(1) = 1$ und $f(n) = f(n-1) + f(n-2)$ für $n \geq 2$

$f(2) = 2,\ f(3) = 3,\ f(4) = 5,\ f(5) = 8,\ f(6) = 13,$

$\ldots,\ f(10) = 89,\ f(11) = 144,\ f(12) = 233, \ldots$

$x_2 := 0;\ x_2 := S(x_2);\ x_3 := x_2;\ x_1 := P(x_1);$

WHILE $x_1 \neq 0$ **BEGIN**

LOOP x_3 **BEGIN** $x_2 := S(x_2)$ **END** ;

$x_4 := x_2;\ x_2 := x_3;\ x_3 := x_4;\ x_1 := P(x_1)$

END ;

$x_1 := x_3$

Erzeugung LOOP/WHILE-berechenbarer Funktionen I

Satz:
Es seien f eine m-stellige **LOOP/WHILE**-berechenbare Funktion und f_i eine n-stellige **LOOP/WHILE**-berechenbare Funktion für $1 \leq i \leq m$. Dann ist auch die n-stellige Funktionen g, die durch

$$g(x_1, x_2, \ldots, x_n) = f(f_1(x_1, \ldots, x_n), f_2(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$$

definiert ist, eine **LOOP/WHILE**-berechenbare Funktion.

Erzeugung LOOP/WHILE-berechenbarer Funktionen II

Satz:
Es seien f und h eine $(n-1)$-stellige bzw. $(n+1)$-stellige **LOOP/WHILE**-berechenbare Funktionen. Dann ist auch die n-stellige Funktionen g, die durch

$$\begin{aligned} g(x_1, x_2, \ldots, x_{n-1}, 0) &= f(x_1, x_2, \ldots, x_{n-1}), \\ g(x_1, x_2, \ldots, x_{n-1}, S(x_n)) &= h(x_1, x_2, \ldots, x_{n-1}, x_n, g(x_1, x_2, \ldots, x_{n-1}, x_n)) \end{aligned}$$

definiert ist, eine **LOOP/WHILE**-berechenbare Funktion.

Erzeugung LOOP/WHILE-berechenbarer Funktionen III

Satz:
Es sei h eine totale $(n+1)$-stellige **LOOP/WHILE**-berechenbare Funktion. Dann ist auch die n-stellige Funktionen g, die durch

$$g(x_1, x_2, \ldots, x_n) = \begin{cases} \min\{m \mid h(x_1, x_2, \ldots, x_n, m) = 0\} & \text{falls } 0 \in rg(h) \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

definiert ist, eine **LOOP/WHILE**-berechenbare Funktion.

Tiefe – Definition

Definition:

Die Tiefe $t(\Pi)$ eines Programms Π wird induktiv wie folgt definiert:

i) Für eine Wertzuweisung Π gilt $t(\Pi) = 1$,

ii) $t(\Pi_1; \Pi_2) = t(\Pi_1) + t(\Pi_2)$,

iii) $t(\mathbf{LOOP}\ x_i\ \mathbf{BEGIN}\ \Pi\ \mathbf{END}) = t(\Pi) + 1$,

iv) $t(\mathbf{WHILE}\ x_i \neq 0\ \mathbf{BEGIN}\ \Pi\ \mathbf{END}) = t(\Pi) + 1$.

Programme kleiner Tiefe I

Programme der Tiefe 1: Wertzuweisungen

Programme der Tiefe 2:

$x_i := A; x_r := B$,

LOOP x_k **BEGIN** $x_i := A$ **END**,

WHILE $x_k \neq 0$ **BEGIN** $x_i := A$ **END**

mit $A \in \{0, x_j, S(x_j), P(x_j)\}$, $B \in \{0, x_s, S(x_s), P(x_s)\}$, $i, j, k, r, s \in \mathbf{N}$

Programme kleiner Tiefe II

Programme der Tiefe 3 sind unter anderem:

$x_i := A'; x_r := B'; x_u := C'$,

$x_i := A'$; **LOOP** x_k **BEGIN** $x_r := B'$ **END**,

$x_i; = A'$; **WHILE** $x_k \neq 0$ **BEGIN** $x_r = B'$ **END**,

LOOP x_k **BEGIN** $x_r := B'$ **END**; $x_i := A'$,

WHILE $x_k \neq 0$ **BEGIN** $x_r = B'$ **END**; $x_i := A'$,

LOOP x_k **BEGIN** $x_i := A'; x_r := B'$ **END**,

WHILE $x_k \neq 0$ **BEGIN** $x_i = A'; x_r := B'$ **END**

mit $A' \in \{0, x_j, S(x_j), P(x_j)\}$, $B' \in \{0, x_s, S(x_s), P(x_s)\}$,
$C' \in \{0, x_v, S(x_v), P(x_v),\ i, j, k, r, s, u, v \in \mathbf{N}$

Nicht-LOOP/WHILE-berechenbare Funktionen

Satz:
Es gibt (mindestens) eine totale Funktion, die nicht **LOOP**/ **WHILE**-berechenbar ist.

Folgerung:
Es gibt eine Funktion f mit folgenden Eigenschaften:
- f ist total,
- der Wertebereich von f ist $\{0, 1\}$,
- f ist nicht **LOOP**/**WHILE**-berechenbar.

Ein spezielles LOOP/WHILE-Programm

$x_1 := S(x_1);\ x_1 := S(x_1);\ x_1 := S(x_1);$
$x_2 := S(x_1);$
LOOP x_1 **BEGIN**
LOOP x_2 **BEGIN** $x_3 := S(x_3)$ **END**
END;
$x_1 := x_3;$
LOOP x_1 **BEGIN**
LOOP x_2 **BEGIN** $x_3 := S(x_3)$ **END**
END;
$x_1 := x_3$

Programm Π' aus ersten sechs Zeilen: $t(\Pi') = 8$ und $\Phi_{\Pi,1}(0,0,0) = 12$
Programm Π aus allen Zeilen: $t(\Pi) = 12$ und $\Phi_{\Pi,1}(0,0,0) = 60$

LOOP – Berechenbarkeit

Definition:
Eine Funktion f heißt **LOOP**-berechenbar, wenn es ein Programm Π mit m Variablen, $m \geq n$, derart gibt, dass in Π keine **WHILE**-Anweisung vorkommt und Π die Funktion f berechnet.

Satz:
Der Definitionsbereich jeder n-stelligen **LOOP**-berechenbaren Funktion ist die Menge $\mathbf{N}^n$, d.h. jede **LOOP**-berechenbare Funktion ist total.

Folgerung:
Die Menge der **LOOP**-berechenbaren Funktionen ist echt in der Menge der **LOOP**/**WHILE**-berechenbaren Funktionen enthalten.

TURING-Maschine – Definition

Definition:
Eine TURING-Maschine ist ein Quintupel

$$M = (X, Z, z_0, Q, \delta),$$

wobei
- X und Z Alphabete sind,
- $z_0 \in Z$ und $\emptyset \subseteq Q \subseteq Z$ gelten,
- δ eine Funktion von $(Z \setminus Q) \times (X \cup \{*\})$ in $Z \times (X \cup \{*\}) \times \{R, L, N\}$ ist, und $* \notin X$ gilt.

TURING-Maschine – Konfiguration

Definition:
Eine Konfiguration K der TURING-Maschine $M = (X, Z, z_0, Q, \delta)$ ist ein Tripel

$$K = (w_1, z, w_2),$$

wobei w_1 und w_2 Wörter über $X \cup \{*\}$ sind und $z \in Z$ gilt.

Eine Anfangskonfiguration liegt vor, falls $w_1 = \lambda$ und $z = z_0$ gelten.

Eine Endkonfiguration ist durch $z \in Q$ gegeben.

TURING-Maschine – Konfigurationsüberführung

Definition:
$M_1 = (w_1, z, w_2)$ und $K_2 = (v_1, z', v_2)$ seien Konfigurationen von M. Wir sagen, dass K_1 durch M in K_2 überführt wird (und schreiben dafür $K_1 \models K_2$), wenn eine der folgenden Bedingungen erfüllt ist:

$$v_1 = w_1,\ w_2 = xu,\ v_2 = x'u,\ \delta(z, x) = (z', x', N)$$

oder

$$w_1 = v,\ v_1 = vx',\ w_2 = xu,\ v_2 = u,\ \delta(z, x) = (z', x', R)$$

oder

$$w_1 = vy,\ v_1 = v,\ w_2 = xu,\ v_2 = yx'u,\ \delta(z, x) = (z', x', L)$$

für gewisse $x, x', y \in X \cup \{*\}$ und $u, v \in (X \cup \{*\})^*$.

TURING-Maschine – induzierte Funktion

Definition:
Sei $M = (X, Z, z_0, Q, \delta)$ eine TURING-Maschine. Die durch M induzierte Funktion f_M aus X^* in X^* ist wie folgt definiert:
$f_M(w) = v$ gilt genau dann, wenn es für die Anfangskonfiguration $K = (\lambda, z_0, w)$ eine Endkonfiguration $K' = (v_1, q, v_2)$, natürliche Zahlen r, s und t und Konfigurationen $K_0, K_1, \ldots, K_t$ derart gibt, daß $*^r v *^s = v_1 v_2$ und

$$K = K_0 \models K_1 \models K_2 \models \ldots \models K_t = K'$$

gelten.

TURING-Maschine – Beispiel 1

$$M_1 = (\{a, b\}, \{z_0, q, z_a, z_b\}, z_0, \{q\}, \delta)$$

δ	z_0	z_a	z_b
$*$	$(q, *, N)$	(q, a, N)	(q, b, N)
a	$(z_a, *, R)$	(z_a, a, R)	(z_b, a, R)
b	$(z_b, *, R)$	(z_a, b, R)	(z_b, b, R)

$$f_{M_1}(x_1 x_2 \dots x_n) = x_2 x_3 \dots x_n x_1$$

TURING-Maschine – Beispiel 2

$$M_2 = (\{a, b\}, \{z_0, z_1, q\}, z_0, \{q\}, \delta),$$

δ	z_0	z_1
$*$	$(z_0, *, N)$	$(q, *, N)$
a	(z_1, a, R)	(z_0, a, R)
b	(z_1, b, R)	(z_0, b, R)

$$f_{M_2}(x_1x_2 \dots x_n) = \begin{cases} x_1x_2 \dots x_n & n \text{ ungerade} \\ nicht\ definiert & \text{sonst.} \end{cases}$$

TURING-Maschine – Beispiel 3

$M_3 = (\{a, b, c, d\}, \{z_0, z_1, z_2, z_3, q, z_a, z_b\}, z_0, \{q\}, \delta)$

δ	z_0	z_1	z_2	z_3	z_a	z_b
$*$	$(z_0, *, N)$	$(z_1, *, N)$	$(z_3, *, L)$			
a	(z_0, a, N)	(z_1, a, N)	(z_2, a, R)	$(z_a, *, L)$	(z_a, a, L)	(z_a, b, L)
b	(z_0, b, N)	(z_1, b, N)	(z_2, b, R)	$(z_b, *, L)$	(z_b, a, L)	(z_b, b, L)
c	(z_1, c, R)	(z_1, c, N)	(z_2, c, N)			
d	(z_0, d, N)	(z_2, d, R)	(z_2, d, N)	(z_3, d, N)	(q, a, N)	(q, b, N)

$$f_{M_3}(w) = \begin{cases} cx_1x_2\ldots x_n & \text{für } w = cdx_1x_2\ldots x_n, x_i \in \{a, b\}, 1 \leq i \leq n, n \geq 1 \\ undefiniert & \text{sonst} \end{cases}$$

TURING-Maschine – Beispiel 4 (Nachfolgerfunktion)

$$M_+ = (\{0,1,2,\ldots,9\}, \{z_0, +, q\}, z_0, \{q\}, \delta)$$

δ	z_0	$+$
$*$	$(+, *, L)$	$(q, 1, N)$
0	$(z_0, 0, R)$	$(q, 1, N)$
1	$(z_0, 1, R)$	$(q, 2, N)$
2	$(z_0, 2, R)$	$(q, 3, N)$
3	$(z_0, 3, R)$	$(q, 4, N)$
4	$(z_0, 4, R)$	$(q, 5, N)$
5	$(z_0, 5, R)$	$(q, 6, N)$
6	$(z_0, 6, R)$	$(q, 7, N)$
7	$(z_0, 7, R)$	$(q, 8, N)$
8	$(z_0, 8, R)$	$(q, 9, N)$
9	$(z_0, 9, R)$	$(+, 0, L)$

Definition der TURING-Berechenbarkeit

Definition:
Eine Funktion $f : X_1^* \to X_2^*$ heißt TURING-berechenbar, wenn es eine TURING-Maschine $M = (X, Z, z_0, Q, \delta)$ derart gibt, dass $X_1 \subseteq X$, $X_2 \subseteq X$ und

$$f_M(x) = \begin{cases} f(x) & \text{falls } f(x) \text{ definiert ist} \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

gelten.

Zwei Sätze über TURING-berechenbare Funktionen

Satz:
Zu jeder TURING-berechenbaren Funktion f gibt es eine TURING-Maschine M, die genau einen Stopzustand hat, stets über dem dem ersten Buchstaben des Ergebnisses stoppt und $f = f_M$ erfüllt.

Lemma:
Sind $f_1 : X^* \to X^*$ und $f_2 : X^* \to X^*$ zwei TURING-berechenbare Funktionen, so ist auch deren Komposition $f : X^* \to X^*$ mit $f(w) = f_2(f_1(w))$ eine TURING-berechenbare Funktion.

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit I

$dec(n)$ – Dezimaldarstellung von n

Satz:
Sei f eine n-stellige **LOOP**/**WHILE**-berechenbare Funktion.
Dann ist die Funktion f', die durch

$$f'(w) = \begin{cases} dec(f(x_1, x_2, \ldots, x_n)) & \text{falls } w = dec(x_1)\#dec(x_2)\#\ldots\#dec(x_n) \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

definiert ist, TURING-berechenbar.

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit II

$M = (X, Z, z_0, Q, \delta)$ – TURING-Maschine
$X \cap Z = \emptyset$ und $X \cup Z = \{a_1, a_2, \ldots, a_p\}$

$\psi : (X \cup Z)^* \to \mathbf{N}$ vermöge
$\psi(a_{i_1} a_{i_2} \ldots a_{i_n}) = \sum_{j=0}^{n} i_j (p+1)^{n-j},\ a_{i_j} \in (X \cup Z)$

ψ – eineindeutige Abbildung von $(X \cup Z)^+$ auf Menge aller natürlichen Zahlen, in deren $(p+1)$-adischer Darstellung keine 0 vorkommt

Satz:
Seien M eine TURING-Maschine und ψ die zugehörige Kodierung. Dann ist die Funktion $f : \mathbf{N} \to \mathbf{N}$ mit $f(\psi(w)) = \psi(f_M(w))$ **LOOP/WHILE**-berechenbar.

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit III

$w = b_1 b_2 \dots b_n$, $b_i \in X \cup Z$ für $1 \leq i \leq n$, $w' \in (X \cup Z)^*$

$Lg(\psi(w)) = |w| = min\{m : (p+1)^m > \psi(w)\}$,

$Prod(\psi(w), \psi(w')) = \psi(ww') = \psi(w)(p+1)^{Lg(\psi(w'))} + \psi(w')$,

$Anfang(\psi(w), i) = \psi(b_1 b_2 \dots b_i) = \psi(w)\ div\ (p+1)^{n-1}$,

$Ende(\psi(w), i) = \psi(b_i b_{i+1} \dots b_n) = \psi(w)\ mod\ (p+1)^{n-i+1}$,

$Elem(\psi(w), i) = \psi(b_i) = Ende(Anfang(\psi(w), i), 1)$

$g(x)$ – erste Position in $x \in (X \cup Z)^*$, an der Element aus Z
– $\min\{i \mid Elem(x, i) \in Z\}$

$r(x) = Anfang(x, g(x) \ominus 2)$,

$s(x) = Prod(Elem(x, g(x) \ominus 1), Elem(x, g(x)), Elem(x, g(x)+1))$,

$t(x) = Ende(x, g(x)+2)$

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit IV

$K = u'azbv'$ mit $a, b \in X$, $u', v' \in X^*$, $z \in Z$
$r(\psi(K)) = \psi(u')$
$s(\psi(K)) = \psi(azb)$,
$t(\psi(K)) = \psi(v')$,

$$\Delta(\psi(K_1)) = \begin{cases} \psi(K_2) & K_1 = azb, a, b \in X, z \in Z, K_1 \models K_2 \\ nicht\ definiert & \text{sonst} \end{cases}$$

$K = u'azbv' = u'K_1v' \models u'K_2v' = K'$
$\psi(K') = Prod(\psi(u'), \Delta(\psi(azb)), \psi(v'))$,
$= Prod(r(K), \Delta(s(K)), t(K))$

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit V

1. Aus w ergibt sich die Anfangskonfiguration $K_0 = z_0 w$ und daraus $\psi(K_0)$.
2. $\overline{\Delta}$ Funktion mit $\overline{\Delta}(\psi(K)) = \psi(K')$ für $K \models K'$
   simuliert einen Überführungsschritt
3. $D(x,n)$ sei definiert durch

$$\begin{aligned} D(x,0) &= x, \\ D(x,n+1) &= \overline{\Delta}(D(x,n)) \end{aligned}$$

   $D(\psi(K),n) = \psi(K'')$, wenn K'' mittels n-facher direkter Überführung aus K entsteht

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit VI

4. h sei Funktion mit

$$h(x) = \begin{cases} 0 & x = \psi(K) \text{ für eine Endkonfiguration } K \\ 1 & \text{sonst} \end{cases}$$

5. $h'(x,n) = h(D(x,n))$,

6. $g(x) = \begin{cases} \min\{n \mid h'(x,n) = 0\} & \text{falls } 0 \in rg(h') \\ \text{nicht definiert} & \text{sonst} \end{cases}$

   liefert Anzahl der Überführungen bis (zum ersten Mal eine) Endkonfiguration erreicht ist

7. $f(x) = D(x, g(x))$
   $f(\psi(K_0))$ liefert $\psi(\overline{K})$ der Endkonfiguration $\overline{K}$

TURING-Berechenbarkeit versus LOOP/WHILE-Berechenbarkeit VII

Satz:
Bis auf eine Konvertierung der Eingabewerte ist eine Funktion genau dann TURING-berechenbar, wenn sie **LOOP**/**WHILE**-berechenbar ist

Folgerung:
Es gibt Funktionen, die nicht TURING-berechenbar sind.

Beschreibung von Entscheidbarkeitsproblemen

Entscheidungsproblem P beschreibbar als

– Aussageform,
d.h. Ausdruck $A_P(x_1, x_2, \ldots, x_n)$ mit einer oder mehreren Variablen x_i,
der bei Ersetzen der Variablen x_i durch Elemente a_i aus dem Grundbereich X_i
in eine Aussage $A_P(a_1, a_2, \ldots, a_n)$ überführt wird, die den Wahrheitswert
"wahr" oder "falsch" annimmt

– durch ein *"Gegeben:"*, d.h. Belegung $a_1, a_2, \ldots, a_n$ der Variablen, und
durch die *"Frage:"* nach der Gültigkeit von $A_P(a_1, a_2, \ldots, a_n)$.

– Menge $M_P = \{(a_1, a_2, \ldots, a_n) : A_P(a_1, a_2, \ldots, a_n)\}$

– Funktion $\varphi_P(x_1, x_2, \ldots, x_n)) = \begin{cases} 1 & (x_1, x_2, \ldots, x_n) \in M_P \\ 0 & \text{sonst} \end{cases}$

Beschreibung des Halteproblems für TURING-Maschinen

$A_P(x, y)$ – x stoppt bei Abarbeitung von y

(wobei x ist mit einer TURING-Maschine und y mit einem Wort zu belegen sind)

Gegeben:	TURING-Maschine M, Wort w
Frage:	Gilt "M stoppt bei Abarbeitung von w" ?
Gegeben:	TURING-Maschine M, Wort w
Frage:	Stoppt M bei Abarbeitung von w ?
Gegeben:	TURING-Maschine M, Wort w
Frage:	Ist $f_M(w)$ definiert?

$M_P = \{(M, w) \mid M \text{ stoppt auf } w\}$

$$\varphi_P(M, w) = \begin{cases} 1 & M \text{ stoppt auf } w \\ 0 & \text{sonst} \end{cases}$$

Algorithmische Entscheidbarkeit

Definition:
Wir sagen, dass ein Problem P algorithmisch entscheidbar (oder kurz nur entscheidbar) ist, wenn die zum Problem gehörende charakteristische Funktion φ_P TURING-berechenbar ist.
Anderenfalls heißt P (algorithmisch) unentscheidbar.

Definition:
Wir sagen, dass eine Menge M (algorithmisch) entscheidbar (oder rekursiv) ist, wenn die zugehörige charakteristische Funktion φ_M TURING-berechenbar ist.
Anderenfalls heißt M (algorithmisch) unentscheidbar.

Problem P genau dann entscheidbar, wenn zugehörige Menge M_P entscheidbar

Berechnungsprobleme

Berechnungsproblem –
für eine Funktion $f : X \to Y$ wird nach dem Wert $f(x)$ gefragt

$M_f = \{(x, y) : f(x) = y\}$

Gegeben: $x \in X$ und $y \in Y$
Frage: Nimmt f an der Stelle x den Wert y an?

$$\varphi_f(x, y) = \begin{cases} 1 & f(x) = y \\ 0 & \text{sonst} \end{cases}$$

Bemerkung:
f ist berechenbar genau dann, wenn φ_f berechenbar
genau dann, wenn M_f entscheidbar ist

Unentscheidbare Probleme I

Satz:
Das Halteproblem für TURING-Maschinen ist unentscheidbar.

Satz:
Das Problem

Gegeben: **LOOP**/**WHILE**-Programm Π, $n \in \mathbf{N}$
Frage: Ist $\Phi_{\Pi,1}(n)$ definiert?

ist unentscheidbar.

Beweis der Unentscheidbarkeit des Halteproblems I

$M = (X, Z, z_0, Q, \delta)$ – TURING-Maschine

$x_0 = *$, $X = \{x_1, x_2, \ldots, x_n\}$,
$Z = \{z_0, z_1, \ldots, z_m\}$, $Q = \{z_{k+1}, z_{k+2}, \ldots, z_m\}$,
$\delta_{ij} = (z_i, x_j, z_{ij}, x_{ij}, r_{ij})$ für $\delta(z_i, x_j) = (z_{ij}, x_{ij}, r_{ij})$, $0 \leq i \leq k$, $0 \leq j \leq n$

Beschreibung von M durch
$x_1, x_2, \ldots, x_n, z_0, z_1, \ldots, z_k, \delta_{00}, \delta_{01}, \ldots, \delta_{0n}, \delta_{10}, \delta_{11}, \ldots, \delta_{1n}, \ldots, \delta_{kn}$

Kodierung:
$x_j \to 01^{j+1}0$ für $0 \leq j \leq n$,
$z_i \to 01^{i+1}0^2$ für $0 \leq i \leq k$,
$R \to 010^3$, $L \to 01^20^3$, $N \to 01^30^3$,
$(\to 010^4$, $) \to 01^20^4$, $, \to 010^5$

Beweis der Unentscheidbarkeit des Halteproblems II

TURING-Maschine M_2

$a\,,\ b\,,\ z_0\,,\ z_1\,,\ (\ z_0\ ,\ *\ ,\ z_0\ ,\ *\ ,\ N\ )\,,\ (\ z_0\ ,\ a\ ,\ z_1\ ,\ a\ ,\ R\ )\,,\ (\ z_0\ ,\ b\ ,\ z_1\ ,\ b\ ,\ R\ )\,,$
$(\ z_1\ ,\ *\ ,\ q\ ,\ *\ ,\ N\ )\,,\ (\ z_1\ ,\ a\ ,\ z_0\ ,\ a\ ,\ R\ )\,,\ (\ z_1\ ,\ b\ ,\ z_0\ ,\ b\ ,\ R\ )$

Kodierung:

$* \to 010, \quad a \to 01^20, \quad b \to 01^30, \quad z_0 \to 010^2, \quad z_1 \to 01^20^2, \quad q \to 01^30^2,$
$R \to 010^3, \quad L \to 01^20^3, \quad N \to 01^30^3, \quad (\to 010^4, \quad) \to 01^20^4, \quad , \to 010^5$

$01^20\ 010^5\ 01^30\ 010^5\ 010^2\ 010^5\ 01^20^2\ 010^5$
$010^4\ 010^2\ 010^5\ 010\ 010^5\ 010^2\ 010^5\ 010\ 010^5\ 01^30^3\ 01^20^4\ 010^5$
$010^4\ 010^2\ 010^5\ 01^20\ 010^5\ 01^20^2\ 010^5\ 01^20\ 010^5\ 010^3\ 01^20^4\ 010^5$
$010^4\ 010^2\ 010^5\ 01^30\ 010^5\ 01^20^2\ 010^5\ 01^30\ 010^5\ 010^3\ 01^20^4\ 010^5$
$010^4\ 01^20^2\ 010^5\ 010\ 010^5\ 01^30^2\ 010^5\ 010\ 010^5\ 01^30^3\ 01^20^4\ 010^5$
$010^4\ 01^20^2\ 010^5\ 01^20\ 010^5\ 010^2\ 010^5\ 01^20\ 010^5\ 010^3\ 01^20^4\ 010^5$
$010^4\ 01^20^2\ 010^5\ 01^30\ 010^5\ 010^2\ 010^5\ 01^30\ 010^5\ 010^3\ 01^20^4$

Beweis der Unentscheidbarkeit des Halteproblems III

S – Menge aller TURING-Maschinen $M = (X, Z, z_0, Q, \delta)$ mit $X = \{0, 1\}$, $Z = \{z_0, z_1, \ldots, z_m\}$, $Q = \{z_m\}$ für ein $m \geq 1$.

w_M – Wort, das $M \in \mathcal{S}$ mittels Kodierung beschreibt

Hilfssatz 1. Das Problem
Gegeben: $w \in \{0, 1\}^*$
Frage: Ist w Kodierung einer TURING-Maschine aus S ?
ist entscheidbar.

$f : \{0, 1\}^* \to \{0, 1\}$ mit

$$f(w) = \begin{cases} 0 & w = w_M \text{ für ein } M \in \mathcal{S},\ f_M(w_M) \text{ ist nicht definiert} \\ nicht\ definiert & \text{sonst} \end{cases}$$

Hilfssatz 2. f ist nicht TURING-berechenbar.

Unentscheidbare Probleme II

Definition:
i) Zwei TURING-Maschinen M_1 und M_2 heißen äquivalent, wenn $f_{M_1} = f_{M_2}$ gilt.
ii) Zwei **LOOP**/**WHILE**-Programme Π_1 und Π_2 heißen äquivalent, wenn $\Phi_{\Pi_1,1} = \Phi_{\Pi_2,1}$ gilt.

Satz:
Das Äquivalenzproblem für TURING-Maschinen bzw. **LOOP**/**WHILE**-Programme ist unentscheidbar.

Unentscheidbare Probleme III

Satz:
Das 10. HILBERTSCHE Problem

Gegeben: eine natürliche Zahl $n \geq 1$, ein Polynom
$p(x_1, x_2, \ldots, x_n) = \sum c_{i_1 i_2 \ldots i_n} x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$
in n Variablen mit ganzzahligen Koeffizienten

Frage: Gibt es eine Lösung von $p(x_1, x_2, \ldots, x_n) = 0$ in $\mathbf{Z}^n$?

ist unentscheidbar.

Unentscheidbare Probleme IV

Satz:
Das Postsche Korrespondenzproblem

Gegeben: Alphabet X mit mind. zwei Buchstaben, $n \geq 1$,
Menge $\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ mit $u_i, v_i \in X^+$ für $1 \leq i \leq n$

Frage: Gibt es eine Folge $i_1 i_2 \ldots i_m$ mit $1 \leq i_j \leq n$ für $1 \leq j \leq m$ derart, dass

$$u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m}$$

gilt?

ist unentscheidbar.

Unentscheidbare Probleme V

Satz:
Das Erfüllbarkeitsproblem der Prädikatenlogik

Gegeben: prädikatenlogischer Ausdruck $H(x_1, x_2, \ldots, x_n)$ über der Signatur $\mathcal{S}$

Frage: Gibt es eine Interpretation von $\mathcal{S}$ und eine Belegung der Variablen $x_1, x_2, \ldots x_n$ derart, dass $H(x_1, x_2, \ldots, x_n)$ wahr wird?

ist unentscheidbar.