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

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

Jede Wertzuweisung ist ein Programm.

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

$\Pi_1; \Pi_2$, $\quad$ **LOOP** x_i **BEGIN** Π **END** , $\quad$ **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 $f : \mathbb{N}_0^n \to \mathbb{N}_0$ 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 ;

$x_1 := x_3$

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

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

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 $\mathbb{N}_0^n$, 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.

Rekursive Funktionen – Basisfunktionen

- die nullstellige Funktion Z, die den konstanten Wert 0 liefert,
- die Funktion $S : \mathbb{N}_0 \to \mathbb{N}_0$, bei der jeder natürlichen Zahl ihr Nachfolger zugeordnet wird,
- die Funktion $P : \mathbb{N}_0 \to \mathbb{N}_0$, bei der jede natürliche Zahl $n \geq 1$ auf ihren Vorgänger und die 0 auf sich selbst abgebildet wird,
- die Funktionen $P_i^n : \mathbb{N}_0^n \to \mathbb{N}_0$, die durch

$$P_i^n(x_1, x_2, \ldots, x_n) = x_i$$

definiert sind.

Rekursive Funktionen – Operationen

- *Kompositionsschema*:
  Für eine m-stellige Funktion g und m n-stellige Funktionen $f_1, f_2, \ldots, f_m$ definieren wir die n-stellige Funktion f vermöge

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

- *Rekursionsschema*:
  Für fixierte natürliche Zahlen $x_1, x_2 \ldots, x_n$, eine n-stellige Funktion g und eine $(n+2)$-stellige Funktion h definieren wir die $(n+1)$-stellige Funktion f vermöge

$$\begin{aligned} f(x_1, x_2, \ldots, x_n, 0) &= g(x_1, x_2, \ldots, x_n), \\ f(x_1, x_2, \ldots, x_n, y+1) &= h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)). \end{aligned}$$

Rekursive Funktionen – Definition und Charakterisierung

Definition:
Eine Funktion $f : \mathbb{N}_0^n \to \mathbb{N}_0$ heißt primitiv-rekursiv, wenn sie mittels endlich oft wiederholter Anwendung von Kompositions- und Rekursionsschema aus den Basisfunktionen erzeugt werden kann.

Satz:
Eine Funktion f ist genau dann primitiv-rekursiv, wenn sie **LOOP**-berechenbar ist.

Primitiv-rekursive Funktionen – Beispiele I

a) f mit $f(n) = S(S(n))$
f' mit $f'(n) = S(f(n)) = S(S(S(n)))$
f bzw. f' ordnen jeder natürlichen Zahl ihren zweiten bzw. dritten Nachfolger zu

b) $P(S(x))$ ist primitiv-rekursiv
$P(S(x)) = x$
$id : \mathbb{N} \to \mathbb{N}$ mit $id(x) = x$ ist primitiv-rekursiv,

Primitiv-rekursive Funktionen – Beispiele II

c) Addition natürlicher Zahlen
$add(x,0) = id(x)$,
$add(x,y+1) = S(P_3^3(x,y,add(x,y)))$

Multiplikation natürlicher Zahlen
$mult(x,0) = Z(x)$,
$mult(x,y+1)$
$= add(P_1^3(x,y,mult(x,y)), P_3^3(x,y,mult(x,y)))$

d) $sum(0) = Z(0) = 0$, $sum(y+1) = S(add(y,sum(y)))$
definieren

$$sum(y) = \sum_{i=0}^{y} i = \frac{y(y+1)}{2}$$

μ-Operator

Für eine $(n+1)$-stellige Funktion h definieren wir die n-stellige Funktion f wie folgt:
$f(x_1, x_2, \ldots, x_n) = z$ gilt genau dann, wenn die folgenden Bedingungen erfüllt sind:
- $h(x_1, x_2 \ldots, x_n, y)$ ist für alle $y \leq z$ definiert,
- $h(x_1, x_2, \ldots, x_n, y) \neq 0$ für $y < z$,
- $h(x_1, x_2, \ldots, x_n, z) = 0$.

Bezeichnung: $f(x_1, \ldots, x_n) = (\mu y)[h(x_1, \ldots, x_n, y) = 0]$, $f = (\mu y)[h]$

Beispiele:

a) $(\mu y)[add(x,y)] = \begin{cases} 0 & \text{für } x = 0 \\ \text{nicht definiert} & \text{sonst} \end{cases}$

b) Für $h(x,y) = |9x^2 - 10xy + y^2|$ gilt $(\mu y)[h(x,y)] = id$

Partiell-rekursive Funktionen

Definition:
Eine Funktion $f : \mathbb{N}_0^n \to \mathbb{N}_0$ heißt partiell-rekursiv, wenn sie mittels endlich oft wiederholter Anwendung von Kompositionsschema, Rekursionsschema und μ-Operator aus den Basisfunktionen erzeugt werden kann.

Satz:
Eine Funktion ist genau dann partiell-rekursiv, wenn sie **LOOP**/**WHILE**-berechenbar ist.

Folgerung:
Es gibt eine totale Funktion, die nicht partiell-rekursiv ist.

Registermaschine – Definition I

i) Eine Registermaschine besteht aus den Registern $B,\ C_0,\ , C_1, C_2, \ldots, C_n, \ldots$ und einem Programm.
B heißt Befehlszähler, C_0 heißt Arbeitsregister oder Akkumulator, und jedes der Register C_n, $n \geq 1$, heißt Speicherregister.

Jedes Register enthält als Wert eine Zahl aus $\mathbb{N}_0$.

ii) Unter einer Konfiguration der Registermaschine verstehen wir das unendliche Tupel $(b, c_0, c_1, \ldots, c_n, \ldots)$, wobei
das Register B die Zahl b enthält,
für $n \geq 0$ das Register C_n die Zahl c_n enthält.

iii) Das Programm ist eine endliche Folge von Befehlen.

Registermaschine – Definition II

Liste der zugelassenen Befehle und der von ihnen bewirkte Änderung der Konfiguration $(b, c_o, c_1, \ldots c_n, \ldots)$ in die Konfiguration $(b', c'_0, c'_1, \ldots, c'_n, \ldots)$ (wobei $u' = u$ für die nicht angegebenen Komponenten gilt)

Ein- und Ausgabebefehle:

`LOAD` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c'_0 = c_i$
`ILOAD` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c'_0 = c_{c_i}$
`CLOAD` i ,	$i \in \mathbb{N}_0$	$b' = b + 1$	$c'_0 = i$
`STORE` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c'_i = c_0$
`ISTORE` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c'_{c_i} = c_0$

Sprungbefehle:

`GOTO` i , $\quad i \in \mathbb{N} \quad b' = i$

`IF` $c_0 = 0$ `GOTO` i , $\quad i \in \mathbb{N} \quad b' = \begin{cases} i & \text{für } c_0 = 0 \\ b+1 & \text{sonst} \end{cases}$

Registermaschine – Definition III

Arithmetische Befehle:

`ADD` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = c_0 + c_i$
`CADD` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = c_0 + i$
`SUB` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - c_i & \text{für } c_0 \geq c_i \\ 0 & \text{sonst} \end{cases}$
`CSUB` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - i & \text{für } c_0 \geq i \\ 0 & \text{sonst} \end{cases}$
`MULT` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = c_0 \cdot c_i$
`CMULT` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = c_0 \cdot i$
`DIV` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / c_i \rfloor$
`CDIV` i ,	$i \in \mathbb{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / i \rfloor$

Stopbefehl:

`END`

Registermaschine – induzierte Funktion

Definition:
Sei M eine Registermaschine. Die von M induzierte Funktion $f_M : \mathbb{N}_0^n \longrightarrow \mathbb{N}_0$ ist wie folgt definiert:
$f(x_1, x_2, \ldots x_n) = y$ gilt genau dann, wenn M ausgehend von der Konfiguration $(1, 0, x_1, x_2, \ldots, x_n, 0, 0, \ldots)$ die Konfiguration $(b, c_0, y, c_2, c_3, \ldots)$ für gewisse $b, c_0, c_2, c_3, \ldots$ erreicht und der b-te Befehl des Programms `END` ist.

Registermaschine – Beispiel 1

Registermaschine M_1 mit Programm

```
 1   CLOAD 1              8   LOAD 2
 2   STORE 3              9   CSUB 1
 3   LOAD 2              10   STORE 2
 4   IF c_0 = 0 GOTO 12  11   GOTO 4
 5   LOAD 3              12   LOAD 3
 6   MULT 1              13   STORE 1
 7   STORE 3             14   END
```

$$f_{M_1}(x,y) = 1 \cdot \underbrace{x \cdot x \cdot \cdots \cdot x}_{y \ mal} = x^y$$

Registermaschine – Beispiel 2

Registermaschine M_2 mit Programm

```
 1   LOAD 1                    8   LOAD 1
 2   IF c_0 = 0 GOTO 12        9   CSUB 1
 3   LOAD 2                   10   STORE 1
 4   CADD 1                   11   GOTO 1
 5   STORE 2                  12   LOAD 3
 6   ADD 3                    13   STORE 1
 7   STORE 3                  14   END
```

$f_{M_2}(n) = \sum_{i=1}^{n} i$

Registermaschinen versus LOOP/WHILE-Programme I

Satz:
Zu jedem **LOOP**/**WHILE**-Programm Π gibt es eine Registermaschine M derart, dass $f_M = \Phi_{\Pi,1}$ gilt.

Beweis: (Induktion über die Tiefe)

$x_i := 0$ wird simuliert durch	$x_i := S(x_j)$ wird simuliert durch
1 CLOAD 0 2 STORE 1 3 END	1 LOAD j 2 CADD 1 3 STORE i 4 END

Registermaschinen versus LOOP/WHILE-Programme II

$\Pi = \Pi_1; \Pi_2$

M_i – Registermaschine mit $f_{M_i} = \Phi_{\Pi_i,1}$, $i \in \{1,2\}$,
Programm P_i von M_i bestehe aus r_i Befehlen
$p_{i,j}$ sei der j-te Befehl von P_i
$p_{i,r_i} =$ `END` und dies einziger Stopp-Befehl in P_i

$q_{j,2} = p_{j,2}$, falls $p_{j,2}$ kein Sprungbefehl
sonst entstehe $q_{j,2}$ aus $p_{j,2}$ durch Erhöhung der Sprungadresse um $r_1 - 1$

1	$p_{i,1}$	r_1	$q_{1,2}$
2	$p_{r_1-1,1}$	r_1+1	$q_{2,2}$
...	...	...	...
r_1-1	$p_{r_1-1,1}$	r_1+r_2-1	$q_{r_2,2}$

berechnet $\Phi_{\Pi,1}$

Registermaschinen versus LOOP/WHILE-Programme III

$\Pi' =$ **WHILE** $x_i \neq 0$ **BEGIN** Π **END**

M – Registermaschine M mit $f_M = \Phi_{\Pi,1}$
$p_1, p_2, \ldots, p_r$ Befehle von M
$p_r =$ `END` einziger Stoppbefehl

q_i entstehe aus p_i durch Erhöhung aller Befehlsnummer um 2 (sowohl Nummern der Befehle als auch Nummern der Sprungadressen)

1	`LOAD` i
2	`IF` $c_0 = 0$ `GOTO` $r+3$
3	q_1
4	q_2
...	...
$r+1$	q_{r-1}
$r+2$	`GOTO 1`
$r+3$	`END`

berechnet $\Phi_{\Pi',1}$

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)

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

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	$(z_0, 9, R)$	$(+, 0, L)$

TURING-Maschine – Normalform und Lemma

Lemma:
Zu jeder TURING-Maschine $M = (X, Z, z_0, Q, \delta)$ gibt es eine TURING-Maschine

$$M' = (X \cup \{\S, \#\}, Z', z_0', \{q'\}, \delta')$$

mit

$$f_{M'}(w) = \begin{cases} f_M(w) & \text{für } w \in X^* \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

derart, dass jede Endkonfiguration von M' die Form (λ, q', v) hat (d.h. die Maschine M' hat genau einen Stopzustand, stoppt nur auf Wörtern über X und stoppt stets über dem ersten Buchstaben des Ergebnisses v).

k-Band-TURING-Maschine – Definition

Definition:
Eine k-Band-TURING-Maschine ist ein 6-Tupel

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

wobei $k \geq 1$ eine natürliche Zahl ist, X, Z, z_0 und Q wie bei einer TURING-Maschine definiert sind, δ eine Funktion

$$(Z \setminus Q) \times (X \cup \{*\})^{k+1} \longrightarrow Z \times (X \cup \{*\})^{k+1} \times \{R, L, N\}^{k+1} \times \{R, N\}$$

ist und $* \notin X$ gilt.

k-Band-TURING-Maschine – Konfiguration

Definition:
Sei M eine k-Band-TURING-Maschine.
Eine Konfiguration von M ist ein $(2k+5)$-Tupel

$$(z, w_e, w'_e, w_1, w'_1, w_2, w'_2, \ldots, w_k, w'_k, w_a, w'_a),$$

wobei $z \in Z$, $w_e, w'_e, w_a, w'_a \in (X \cup \{*\})^*$ und $w_i, w'_i \in (X \cup \{*\})^*$ für $1 \leq i \leq k$ gelten.

Eine Konfiguration heißt Anfangskonfiguration, falls $z = z_0$, $w_e = w_a = w_1 = w_2 = \cdots = w_k = \lambda$ und $w'_a = w'_1 = w'_2 = \cdots = w'_k = *$ gelten.

Eine Konfiguration heißt Endkonfiguration, falls z in Q liegt.

k-Band-TURING-Maschine – induzierte Funktion

Definition:
Sei M eine k-Band-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

$$K = (z_0, \lambda, w, \lambda, *, \lambda, *, \ldots, \lambda, *)$$

eine Endkonfiguration

$$K' = (q, w_e, w'_e, w_1, w'_1 w_2, w'_2, \ldots, w_k, w'_k, w_a, w'_a),$$

und Konfigurationen $K_0, K_1, \ldots, K_t$ derart gibt, dass

$$K = K_0 \models K_1 \models K_2 \models \ldots \models K_t = K' \text{ und } v = w_a w'_a$$

gelten.

k-Band-TURING-Maschine – Beispiel

$M = (2, \{a, b\}, Z = \{z_0, z_1, z_2, q\}, z_0, Q = \{q\}, \delta)$

(a1) $\delta(z_0, x, *, *) = (z_0, x, x, *, R, R, R, N)$ für $x \in X$,
(a2) $\delta(z_0, *, *, *) = (z_1, *, *, *, N, L, L, N)$,
(a3) $\delta(z_1, *, x, y) = (z_1, x, y, *, N, L, N, N)$ für $x, y \in X$,
(a4) $\delta(z_1, *, *, y) = (z_2, *, y, *, N, R, N, N)$ für $y \in X$,
(a5) $\delta(z_2, *, x, x) = (z_2, x, x, *, N, R, L, N)$ für $x \in X$,
(a6) $\delta(z_2, *, x, y) = (q, x, y, b, N, N, N, N)$ für $x, y \in X,\ x \neq y$,
(a7) $\delta(z_2, *, *, *) = (q, *, *, a, N, N, N, N)$,
(a8) $\delta(z, x, y, v) = (z, y, v, *, N, N, N, N)$ sonst

$$f_M(w) = \begin{cases} a & w \text{ ist Palindrom} \\ b & \text{sonst} \end{cases}$$

TURING-Berechenbarkeiten

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

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

gibt.

Satz:
Zu jeder k-Band-TURING-Maschine M gibt es eine TURING-Maschine M' derart, dass $f_{M'} = f_M$ gilt.

Registermaschinen versus k-Band-TURING-Maschinen

$dec(n)$ – Dezimaldarstellung von n

Satz:
Sei M eine Registermaschine M mit $f_M : \mathbb{N}_0^n \to \mathbb{N}_0$. Dann gibt es eine 3-Band-TURING-Maschine M', deren Eingabealphabet außer den Ziffern 0,1,2,. . . 9 noch das Trennsymbol $\#$ und das Fehlersymbol F enthält und deren induzierte Funktion

$$f_{M'}(w) = \begin{cases} dec(f_M(m_1, m_2, \ldots, m_n)) & w = dec(m_1)\#dec(m_2)\ldots\#dec(m_n) \\ F & \text{sonst} \end{cases}$$

gilt (auf einer Eingabe, die einem Zahlentupel entspricht, verhält sich M' wie M und gibt bei allen anderen Eingaben eine Fehlermeldung).

TURING-Maschinen versus partiell-rekursive Funktionen I

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

$\psi : (X \cup Z)^* \to \mathbb{N}$ 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 $f : \mathbb{N} \to \mathbb{N}$ mit $f(\psi(w)) = \psi(f_M(w))$ partiell-rekursiv.

TURING-Maschinen versus partiell-rekursive Funktionen II

$w = b_1 b_2 \dots b_n$, $b_i \in X \cup Z$ 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-Maschinen versus partiell-rekursive Funktionen III

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

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

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

TURING-Maschinen versus partiell-rekursive Funktionen IV

1. Aus w ergibt sich die Anfangskonfiguration $K_0 = z_0 w$.
2. $\overline{\Delta}$ Funktion mit $\overline{\Delta}(\psi(K)) = \psi(K')$ für $K \models K'$

   D wird 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

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

   $h'(x,y) = h(D(x,y))$,
   $(\mu y)[h']$ liefert Anzahl der Überführungen bis Endkonfiguration erreicht ist
   $D(\psi(K_0),(\mu y)[h'])$ liefert $\psi(\overline{K})$ der Endkonfiguration $\overline{K}$
3. Aus $\psi(\overline{K})$ ermitteln wir das Wort auf dem Band.

Zusammenfassung

Satz:
Für eine Funktion f sind die folgenden Aussagen gleichwertig:

- f ist durch ein **LOOP**/**WHILE**-Programm berechenbar.
- f ist partiell-rekursiv.
- f ist durch eine Registermaschine berechnenbar.
- f ist bis auf Konvertierung der Zahlendarstellung durch eine TURING-Maschine berechenbar.
- f ist bis auf Konvertierung der Zahlendarstellung durch eine k-Band-TURING-Maschine berechenbar.

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 $A_P(a_1, \ldots, a_n)$ 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, $n \in \mathbb{N}$
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

$\mathcal{S}$ – 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 mindestens 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.