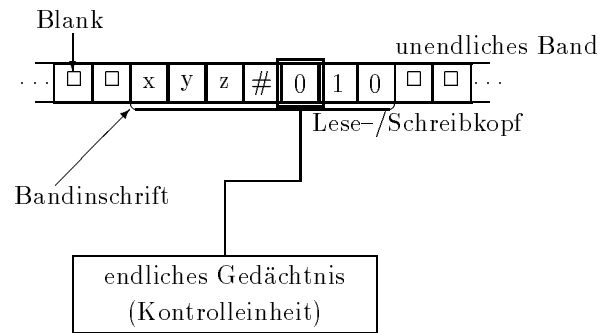


1.3 Turing-Berechenbarkeit

Turings Vorschlag war, den Berechenbarkeitsbegriff mit Hilfe einer sehr einfachen Rechenmaschine (heute Turing-Maschine genannt) zu erfassen.



Definition.

Eine (Nichtdeterministische) Turingmaschine (kurz TM) ist ein 7-Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, E)$$

mit

Q endliche Menge von “Zuständen”

Σ endliche Menge von “Eingabesymbolen” (Eingabealphabet)

$\Gamma \supset \Sigma$ endliche Menge von “Bandsymbolen” (Arbeitsalphabet)

$$\left. \begin{array}{l} \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\} \text{ “deterministische”} \\ \delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R, N\}} \text{ “nichtdeterministische”} \end{array} \right\} \text{Überföhrungsfunktion}$$

$q_0 \in Q$ “Startzustand”

$\square \in \Gamma - \Sigma$ “Blank” (leeres Bandsymbol)

$E \subseteq Q$ endliche Menge von “Endzuständen”

Informal bedeutet

$$\delta(q, a) = (q', b, m) \text{ bzw. } \delta(q, a) \ni (q', b, m)$$

folgendes:

Befindet sich M in Zustand q und liest Bandsymbol a (unter Lese-/Schreibkopf), so geht M im nächsten Schritt in Zustand q' über, überschreibt a durch b und

führt Kopfbewegung $m \in \{L(inks), R(rechts), N(ichts)\}$ aus.

Definition.

Eine Konfiguration einer TM M ist ein Wort $k \in \Gamma^* Q \Gamma^*$

Konfigurationen sind “Momentaufnahmen” von M :

$k = \alpha q \beta$ bedeutet:

$\alpha \beta$ ist der nichtleere (bzw. besuchte) Teil der Bandinschrift.

q ist der Zustand der TM .

Das erste Zeichen von β ist das Zentrum des Lese-/Schreibkopfes.

Startkonfiguration bei Eingabe $w \in \Sigma^* : q_0 w$

$\square \dots \square w \square \dots \square$
 $\uparrow$

w steht auf dem Band

Der Lese-/Schreibkopf steht auf dem ersten Symbol von w

M ist in Zustand q_0

Formale Beschreibung der Arbeit einer TM :

Definition.

Beschreibe die Relation $\vdash$ auf der Menge der Konfigurationen von M :

$$a_1 \dots a_m q b_1 \dots b_n \vdash \begin{cases} a_1 \dots a_m q' c b_2 \dots b_n & \text{falls } (q', c, N) \in \delta(q, b_1) \ m \geq 0, n \geq 1 \\ a_1 \dots a_m c q' b_2 \dots b_n & \text{falls } (q', c, R) \in \delta(q, b_1) \ m \geq 0, n \geq 2 \\ a_1 \dots a_{m-1} q' a_m c b_2 \dots b_n & \text{falls } (q', c, L) \in \delta(q, b_1) \ m \geq 1, n \geq 1 \\ a_1 \dots a_m \square q' c & \text{falls } (q', c, R) \in \delta(q, b_1) \text{ und } n = 1 \\ q' \square c b_2 \dots b_n & \text{falls } (q', c, L) \in \delta(q, b_1) \text{ und } m = 0 \end{cases}$$

$\vdash^*$ bezeichne den transitiven Abschluß von $\vdash$

Beispiel.

Eine TM , die die Eingabe $w \in \Sigma^*$ als Binärzahl interpretiert und 1 hinzuaddiert:

$$M = (\{q_0, q_1, q_2, q_e\}, \{0, 1\}, \{0, 1, \square\}, \delta, q_0, \square, \{q_e\})$$

mit

$$\delta(q_0, 0) = (q_0, 0, R)$$

$$\delta(q_0, 1) = (q_0, 1, R)$$

$$\delta(q_0, \square) = (q_1, \square, L)$$

$$\delta(q_1, 0) = (q_2, 1, L)$$

$$\begin{aligned}\delta(q_1, 1) &= (q_1, 0, L) \\ \delta(q_1, \square) &= (q_e, 1, N)\end{aligned}$$

$$\begin{aligned}\delta(q_2, 0) &= (q_2, 0, L) \\ \delta(q_2, 1) &= (q_2, 1, L) \\ \delta(q_2, \square) &= (q_e, \square, R)\end{aligned}$$

Beispiel.

$w=101$

$$q_0 \vdash 1q_001 \vdash 10q_01 \vdash 101q_0\square \vdash 10q_1\square \vdash 1q_100\square \vdash q_2110\square \vdash q_2\square110\square \vdash \square q_e110$$

Definition.

Eine Funktion $f : \mathbb{N}^* \rightarrow \mathbb{N}$ heißt *Turing-berechenbar*, falls es eine (deterministische) TM M gibt, so daß für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \dots, n_k) = m$$

$\iff$

$$q_0 \text{bin}(n_1) \# \text{bin}(n_2) \# \dots \# \text{bin}(n_k) \vdash^* q_e \text{bin}(m)$$

wobei $q_e \in E$.

($\text{bin}(n)$ bezeichnet die Binärdarstellung von $n \in \mathbb{N}$ ohne führende Nullen)

Definition.

Eine Funktion $f : \Sigma^* \rightarrow \Sigma$ heißt *Turing-berechenbar*, falls es eine (deterministische) TM M gibt, so daß für alle $x, y \in \Sigma^*$ gilt:

$$f(x) = y$$

$\iff$

$$q_0 x \vdash^* q_e y$$

wobei $q_e \in E$.

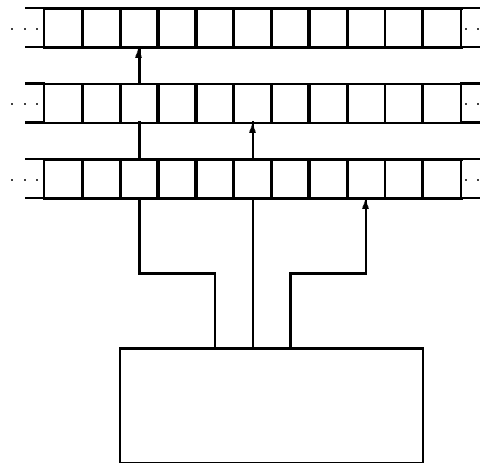
Damit ist ausgedrückt, daß im Falle $f(x) = \text{undef.}$ M in eine unendliche Schleife geht.

Beispiel.

1. $f : n \rightarrow n + 1$ ist Turing-berechenbar.
Das Beispiel überführt $\text{bin}(n)$ in $\text{bin}(n + 1)$
2. die überall nichtdefinierte Funktion ist Turing-berechenbar;
z.B.: durch eine TM mit

$$\delta(q_0, a) = (q_0, a, R) \text{ für alle } a \in \Gamma$$

3. Typ 0- (semientscheidbare) Sprachen sind berechenbar.

Mehrband-TM:

Die Lese-/Schreibköpfe können sich unabhängig voneinander bewegen.

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, N\}^k$$

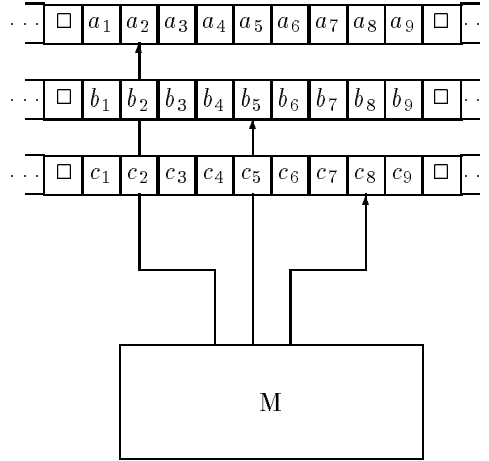
Satz.

Zu jeder Mehrband-TM M gibt es eine 1-Band TM M' die dieselbe Funktion berechnet wie M .

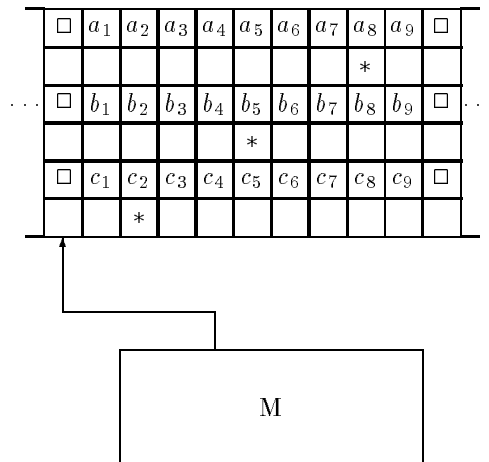
Beweis.

Sei k die Anzahl der Bänder, Γ Arbeitsalphabet von M .

Idee:



wird simuliert durch:



Formal

$$\Gamma' := (\Gamma \cup \{*\})^{2k}$$

M' simuliert M wie folgt:

Gestartet mit Eingabe $x_1, \dots, x_n \in \Gamma^*$ erzeugt M' die Darstellung der Startkonfiguration von M in Spuren-Darstellung.

M' simuliert jeweils einen Schritt von M durch mehrere Schritte:
 M' positioniert den Lese-/Schreibkopf links von allen $*$ -Markierungen.
 M' geht nach rechts bis alle k Markierungen überschritten sind und
 "weiß" nun worauf die δ -Funktion von M anzuwenden ist.
 M' geht in den entsprechenden Zustand über.
 (denn $|Q'| \geq |Q \times \Gamma^k|$)

Man kann in Zukunft einfach eine Mehrbandmaschine angeben, und wissen, daß diese durch eine 1-Band Maschine simuliert werden kann.

- Sei M eine 1-Band TM .

$$M(i, k) ; i \leq n$$

bezeichne die k -Band TM , die aus M dadurch entsteht, daß alle Aktionen auf Band i ablaufen.

Beispiel.

$M : \delta(q, a) = (q', b, y) , y \in \{L, R, N\}$ entspreche den Übergang in M'

$$\delta'(q, c_1, c_2, a, c_3, c_4) = (q', c_1, c_2, b, c_3, c_4, N, N, y, N, N)$$

falls k unwichtig ist, dann schreibe lediglich $M(i)$ statt $M(i, k)$.

- "Hintereinanderschaltung" von TM :
 Seien $M_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i, \square, F_i) ; i = 1, 2 ; Q_1 \cap Q_2 = \emptyset$

$$\text{start} \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \text{stop}$$

bzw.

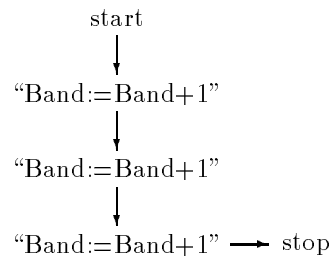
$$M_1; M_2$$

bezeichnet die TM :

$$\begin{aligned}
 M &= (Q_1 \cup Q_2, \Sigma, \Gamma_1 \cup \Gamma_2, \delta, q_1, \square, F_2) \\
 \delta &= \delta_1 \cup \delta_2 \cup \{(q_e, a, q_2, a, N) : q_e \in F_1, a \in \Gamma_1\}
 \end{aligned}$$

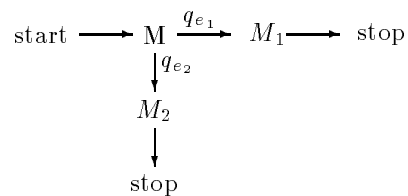
Beispiel.

•



bezeichnet die TM, die 3 hinzuaddiert.

•



bezeichnet die TM, die im Endzustand q_{e_1} M_1 startet und im Endzustand q_{e_2} M_2

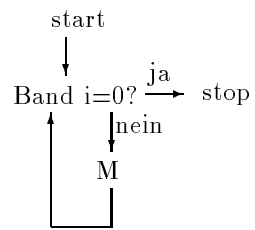
- Betrachte: Die TM “Band=0?”:

$$Q = \{q_0, q_1, \text{ja}, \text{nein}\} \quad q_0 \text{ Startzustand}$$

$$\begin{aligned}
 \delta(q_0, a) &= (\text{nein}, a, N) \text{ für } a \neq \square \\
 \delta(q_0, 0) &= (q_1, 0, R) \\
 \delta(q_1, a) &= (\text{nein}, a, L) \text{ für } a \neq \square \\
 \delta(q_1, \square) &= (\text{ja}, \square, L)
 \end{aligned}$$

Schreibe “Band i=0?” anstelle von “Band=0?”

- “WHILE Band $i \neq 0$ DO M”



1.4 LOOP-, WHILE- und GOTO-Berechenbarkeit

Die einfache Programmiersprache LOOP:

- Variablen: $x_0, x_1, \dots$
- Konstanten: $0, 1, 2, \dots$
- Trennsymbole: $;$, $:=$
- Operationszeichen: $+$, $-$
- Schlüsselwörter: LOOP, DO, END

Induktive Definition der Syntax von LOOP:

- Wertzuweisung

$$\begin{aligned} x_i &:= x_j + c \\ x_i &:= x_j - c, \quad c \text{ Konstante} \end{aligned}$$

ist ein LOOP-Programm

- Sind P_1, P_2 LOOP-Programme, dann auch $P_1; P_2$
- Ist P ein LOOP-Programm, x_i Variable, dann auch

LOOP x_i DO P END;

Semantik der LOOP-Programme:

Soll ein LOOP-Programm eine k -stellige Funktion berechnen, und ist $(n_1, \dots, n_k)$ das Argument, dann gilt:

$$x_i := \begin{cases} n_i & i \leq k \quad (\text{Startsituation}) \\ 0 & \text{sonst} \end{cases}$$

Die Wertzuweisung wird wie üblich interpretiert:

$$x_i := x_j + c.$$

Bei

$$x_i := x_j - c$$

modifizierte Substitution

$$x_i := \begin{cases} x_i - c & c \leq x_i \\ 0 & \text{sonst} \end{cases}$$

$P_1; P_2$ wird so interpretiert, daß zuerst P_1 und dann P_2 ausgeführt wird.

LOOP x_i DO P END wird so interpretiert, daß P sooft ausgeführt wird, wie der Wert der Variable x_i zu Beginn angibt.

Das Resultat ergibt sich als Wert von x_0 .

Definition.

Eine Funktion $f : N^k \rightarrow N$ heißt *LOOP-berechenbar*, falls es ein LOOP-Programm P gibt, das gestartet mit $n_1, \dots, n_k$ in den Variablen $x_1, \dots, x_k$ mit dem Wert $f(n_1, \dots, n_k)$ in x_0 stoppt.

Bemerkung

Alle LOOP-berechenbaren Funktionen sind total definiert.

- IF $x = 0$ THEN A END

kann simuliert werden durch

```

y := 1;
LOOP x DO y := 0 END;
LOOP y DO A END

```

Beispiel.

- Die Addition " $x_0 := x_1 + x_2$ " ist LOOP-berechenbar:

```

x_0 := x_1;
LOOP x_2 DO x_0 := x_0 + 1 END

```

mit Verallgemeinerung auf " $x_0 := x_i + x_j$ "

- Die Multiplikation " $x_0 := x_1 * x_2$ " ist LOOP-berechenbar:

```

x_0 := 0;
LOOP x_2 DO x_0 := x_0 + x_1 END

```

Man kann auch DIV, MOD durch LOOP-Programme beschreiben
 $\leadsto x := (y \text{ DIV } z) + (x \text{ MOD } 5) * z$

Erweiterung der LOOP-Programme durch die WHILE-Schleife
 Ergebnis: WHILE-Programme:

- Ist P ein WHILE-Programm, x_i Variable, dann ist auch

WHILE $x \neq 0$ DO P END;

ein WHILE-Programm.

Semantik.

P wird solange ausgeführt, wie $x_i \neq 0$ gilt.

Man kommt in WHILE-Programmen ohne LOOP aus:

LOOP x DO P END;

wird simuliert durch

$y := x;$
WHILE $y \neq 0$ DO $y := y - 1; P$ END;

Definition.

Eine Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N}$ heißt *WHILE-berechenbar*, falls es ein WHILE-Programm P gibt, das gestartet mit $n_1, \dots, n_k$ in $x_1, \dots, x_k$ (0 sonst) mit dem Wert $f(n_1, \dots, n_k)$ in x_0 stoppt.

Sonst stoppt P nicht.

Satz.

TM können WHILE-Programme simulieren.

D.h. jede WHILE-berechenbare Funktion ist auch TM-berechenbar.

Beweis.

Man kann die Wertzuweisungen, Sequenzen und WHILE-Schleifen mit einer Mehrband TM simulieren.

(Wobei das i -te Band der i -ten Variablen (binär) entspricht.)

Man kann eine Mehrband TM mit einer 1-Band TM simulieren. ■

Beweis der Umkehrung:

Betrachte GOTO-Programme.

GOTO-Programme bestehen aus Folgen von markierten Anweisungen

$M_1 : A_1 ; M_2 : A_2 ; \dots ; M_k : A_k$

Als Anweisungen sind zugelassen:

Wertzuweisungen: $x_i := x_j \pm c$

unbedingter Sprung: GOTO M_i

bedingter Sprung: IF $x_i = c$ THEN GOTO M_i