

3. Turingmaschinen



Alan Mathison Turing (1912 - 1954)

Britischer Logiker, Mathematiker und Computerpionier

<http://www.time.com/time/time100/scientist/profile/turing.html>

FORMALISIERUNG VON ALGORITHMEN

Wegen der beobachteten Zusammenhänge zwischen Berechnungs-, Entscheidungs- und Aufzählungsverfahren genügt es Berechnungsverfahren zu formalisieren. Weiter genügt es Verfahren zur Berechnung von Funktionen über den natürlichen Zahlen zu betrachten.

Wir werden 3 unterschiedliche Ansätze vorstellen:

- Turingmaschinen
- Registermaschinen
- Rekursive Funktionen

Wir werden zeigen, dass diese Ansätze zu demselben formalen Berechenbarkeitsbegriff führen (ÄQUIVALENZSATZ). Man geht davon aus, dass diese formalen Ansätze den intuitiven Berechenbarkeitsbegriff adäquat (d.h. korrekt und vollständig) wiedergeben:

CHURCH-TURING-THESE: Eine Funktion ist genau dann (im intuitiven Sinne) berechenbar, wenn sie von einer Turingmaschine berechnet wird.

UNTERSCHIEDUNGSMERKMALE DER ANSÄTZE:

TURINGMASCHINEN: Diese Maschinen operieren auf Wörtern (d.h. benutzen Zahldarstellungen). Grundoperationen sind elementare Zeichenmanipulationen (Lesen, Streichen, Hinzufügen, Ersetzen einzelner Buchstaben).

REGISTERMASCHINEN: Es wird von der Zahldarstellung abstrahiert. Grundoperationen sind elementare Zähloperationen (Inkrementieren, Dekrementieren, Nulltest).

REKURSIVE FUNKTIONEN: Ausgehend von elementaren berechenbaren Funktionen erzeugt man komplexere berechenbare Funktionen mit Hilfe von effektiven Operationen, die berechenbare Funktionen in berechenbare Funktionen überführen (z.B. Einsetzung, Rekursion).

Die beiden ersten Ansätze sind also *imperativ* (wobei der zweite Ansatz die Idee des *Datentyps* einbezieht), der dritte Ansatz *funktional*.

Algorithmen sind Vorschriften, die die Lösung eines (mathematischen) Problems so exakt beschreiben und in elementare Teilschritte zerlegen, dass die Ausführung rein schematisch erfolgen kann und keinerlei Kreativität erfordert. Im Prinzip kann also der Algorithmus von einer Maschine ausgeführt werden. Eine Turingmaschine ist solch eine 'mathematische' (d.h. formal spezifizierte aber nicht physikalisch realisierte) Maschine, die der Arbeitsweise eines Mathematikers nachempfunden ist und die von diesem bei der Problemlösung verwendeten elementaren Einzelschritte in geeignet standardisierter Form durchführen kann.

Bevor wir die Turingmaschine beschreiben, betrachten wir zunächst die grundlegenden Funktionseinheiten jeder mathematischen Maschine und zeigen dann, wie diese bei einer Turingmaschine konzipiert sind.

Mathematische Maschine = Basismaschine + Programm

Basismaschine (*hardware*):

- Speicherstruktur
- Endlicher Satz von Speicheroperationen
 - Speichertransformationen = Aktualisierung des Speicherinhalts
(z.B. Schreiben in den Speicher)
 - Speichertests = Überprüfen des Speicherinhalts
(z.B. Lesen aus dem Speicher)
- Ein-/Ausgabemechanismus

Programm = Ablaufsteuerung (*software*):

- Festlegung der auszuführenden Folge von Operationen

DIE TURINGMASCHINE

Anschauliche Beschreibung der Komponenten

SPEICHER (STRUKTUR, OPERATIONEN)

- Nach beiden Seiten hin unendliches, in **Felder** aufgeteiltes **Band**.
- Jedes Feld ist mit einem Buchstaben aus dem **Bandalphabet** Γ beschrieben.
(Leere Felder sind mit dem **Blank** b (**Leerzeichen**) beschriftet.)
- Der Zugriff auf die Daten erfolgt mit einem **Lese/Schreibkopf**, der auf einem Feld (dem **Arbeitsfeld**) sitzt und dessen Inschrift lesen und/oder überschreiben kann.
Danach kann der Kopf um ein Feld (nach links oder rechts) verlegt werden.

In einem Rechenschritt wird

- (1) die Inschrift des Arbeitsfeldes gelesen
- (2) die Inschrift des Arbeitsfeldes überschrieben
(eventuell mit der alten Inschrift)
- (3) das Arbeitsfeld (um ein Feld nach links oder rechts) verlegt
(oder beibehalten)

EIN- UND AUSGABE

Ein- und Ausgabe sind Wörter über festgelegtem **Eingabe-** bzw. **Ausgabealphabet**. (Diese Alphabete sind Teil des Bandalphabets.)

EINGABE

- Die Eingabe wird rechts des Arbeitsfeldes auf das ansonsten leere Band geschrieben.
(Dabei werden mehrere Eingaben durch Blanks getrennt.)

Da die Turingmaschine auf Wörtern operiert, müssen Zahleingaben durch ihre Darstellungen ersetzt werden. Wir benutzen hierbei für die Zahl n die Unärdarstellung $\underline{n} = 0^{n+1}$.

AUSGABE

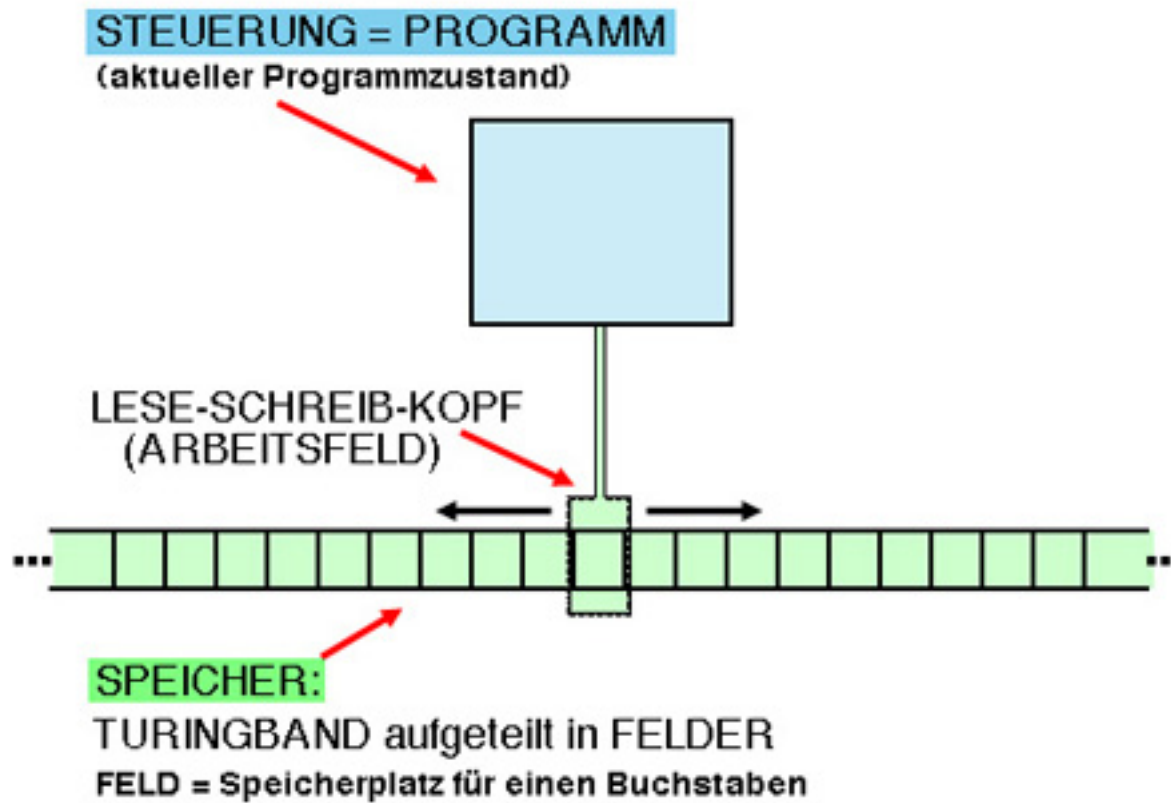
- Die Ausgabe wird dem Band rechts des Arbeitsfeldes entnommen.

Genauer: Die Ausgabe ist das längste Wort über dem Ausgabealphabet, das direkt rechts des Arbeitsfeldes steht.

STEUERUNG (PROGRAMM)

- Die Maschine befindet sich zu jedem Zeitpunkt in einem von endlich vielen möglichen (Programm-)Zuständen.
Zu Beginn der Rechnung ist die Maschine in einem ausgezeichneten Startzustand.
- Die Rechenschritte der Maschine werden durch eine endliche Folge von Instruktionen, dem Programm, festgelegt.
- Eine Instruktion ist dabei eine bedingte Anweisung der folgenden Form:
Falls die Maschine im Zustand z ist und der Buchstabe a auf dem Arbeitsfeld steht, so überdrucke das Arbeitsfeld mit dem Buchstaben a' , bewege den LS-Kopf nach links (oder rechts oder lasse ihn stehen) und gehe in den neuen Zustand z' .
- Zu jeder Bedingung (Paar aus Zustand und Inschrift des Arbeitsfeldes) gibt es höchstens eine zutreffende Instruktion (\rightarrow Determinismus).
Das Programm lässt sich daher auch als eine partielle Funktion
$$\delta : Z \times \Gamma \rightarrow \Gamma \times \text{Bew} \times Z$$
darstellen.
- Lässt sich keine Instruktion anwenden, ist die Rechnung beendet.

SCHEMATISCHE DARSTELLUNG



BEISPIEL

Eine Turingmaschine M zur Berechnung der 2-st. Addition arbeitet wie folgt:

EINGABE: Die Eingaben m und n sind zu Beginn der Rechnung in der Unärdarstellung auf das Band geschrieben:

$$\dots b b \underline{b} 0^{m+1} b 0^{n+1} b b \dots$$

(Hierbei ist das Blank auf dem Arbeitsfeld unterstrichen.)

RECHNUNG: Die Maschine verlegt zunächst den LS-Kopf auf das Blank zwischen den beiden Eingaben, überschreibt dieses durch eine 0, läuft vor den verschmolzen Null-Block $0^{m+1+1+n+1} = 0^{m+n+3}$ zurück, ersetzt die ersten beiden Nullen durch Blanks, lässt den LS-Kopf vor dem verbleibenden Nullblock 0^{m+n+1} stehen und stoppt.

ZUSTÄNDE: Die Phasen der Rechnung werden durch folgende Zustände beschrieben:

- z_0 Startzustand: Gehe nach rechts auf erste Null von 0^{m+1}
- z_{\rightarrow} Durchlaufe 0^{m+1} bis zum Blank dahinter und ersetze dieses durch 0
- z_{\leftarrow} Laufe auf die erste Null von 0^{m+n+3} zurück
- z_1 Ersetze die erste Null durch Blank und gehe auf die zweite Null
- z_2 Ersetze die zweite Null durch Blank
- z_{stop} Stoppe

PROGRAMM: Die Instruktionen von M lassen sich durch folgende Tabelle beschreiben:

Z	\times	Γ	\rightarrow	Γ	\times	Bew	\times	Z
z_0		b		b		R		z_{\rightarrow}
z_{\rightarrow}		0		0		R		z_{\rightarrow}
z_{\rightarrow}		b		0		L		z_{\leftarrow}
z_{\leftarrow}		0		0		L		z_{\leftarrow}
z_{\leftarrow}		b		b		R		z_1
z_1		0		b		R		z_2
z_2		0		b		S		z_{stop}

Das Bandalphabet besteht neben der Null (0) nur aus dem Blank (b).

Die Bewegungen des LS-Schreibkopfes sind:

L = links

R = rechts

S = stehen lassen

(ENDE BEISPIEL)

FORMALE SPEZIFIKATION EINER TURINGMASCHINE M
zur Berechnung einer Funktion $f : (\Sigma^*)^m \rightarrow T^*$:

$$M = (\Sigma, m, T, \Gamma, Z, z_0, \delta)$$

Komponenten von M :

- Σ ist das *Eingabealphabet* und T ist das *Ausgabealphabet*,
- m ist die *Stelligkeit* der zu berechnenden Funktion,
- Γ ist ein Alphabet mit $\Sigma \cup T \subset \Gamma$ und $b \in \Gamma - (\Sigma \cup T)$, das *Bandalphabet*, wobei b das *Leerzeichen (Blank)* ist,
- Z ist eine endliche Menge, genannt die *Zustandsmenge*,
- z_0 ist ein Element aus Z , genannt der *Startzustand*,
- δ ist eine partielle Funktion

$$\delta : Z \times \Gamma \rightarrow \Gamma \times \text{Bew} \times Z,$$

genannt das *Programm* oder die *Übergangsfunktion*, wobei $\text{Bew} = \{L, S, R\} = \{-1, 0, +1\}$ die Menge der *Bewegungen* ist.

BEISPIEL:

Die bereits eingeführte Turingmaschine M zur Berechnung der Addition hat folgende Spezifikation:

$$M = (\{0\}, 2, \{0\}, \{0, b\}, Z, z_0, \delta),$$

wobei

$$Z = \{z_0, z_{\rightarrow}, z_{\leftarrow}, z_1, z_2, z_{stop}\}$$

und δ durch die bereits gegebene Tabelle definiert ist.

BEISPIEL:

Folgende Turingmaschine $M = (\{0\}, 1, \{0\}, \{0, b\}, Z, z_0, \delta)$ berechnet die Funktion $f(n) = 2n$ (bzgl. der Unärdarstellung $\underline{n} = 0^{n+1}$):

Zustände: $Z = \{z_0, z_{begincopy}, z_{\rightarrow}, z_{\rightarrow\rightarrow}, z_{00}, z_{\leftarrow}, z_{fertig?}, z_{\leftarrow\leftarrow}, z_{fertig!}, z_{streiche0}, z_{stop}\}$

Programm: s. nächste Folie

Idee: Mit Hilfe eines iterativen Verfahrens wird der Ausgabeblock $A (=0^{2n+1})$ rechts hinter dem Eingabeblock E (getrennt durch ein Blank) erzeugt, wobei E gleichzeitig von links nach rechts abgebaut wird. Für jede gelöschte Null in E werden dabei zwei neue Nullen rechts an A angehängt, sodass am Ende noch die erste 0 von A gelöscht werden muss.

Im Einzelnen geht man wie folgt vor: Zunächst wird auf die erste 0 von E gegangen ($z_{begincopy}$), diese gelöscht und auf das Blank rechts hinter E gegangen (z_{\rightarrow}), dann weiter hinter den aktuellen A -Block und zunächst eine 0 angefügt ($z_{\rightarrow\rightarrow}$) und dann eine weitere 0 (z_{00}). Es wird dann ans rechte Ende von A zurückgelaufen (z_{\leftarrow}) und geprüft, ob A schon leer ist ($z_{fertig?}$). Ist dies nicht der Fall, wird an den linken Anfang von A gegangen ($z_{\leftarrow\leftarrow}$) und der Kopierzyklus iteriert ($z_{begincopy}$). Andernfalls ($z_{fertig!}$) wird die erste 0 von A noch gestrichen ($z_{streiche0}$) und gestoppt (z_{stop}).

Programm δ :

Z	\times	Γ	\rightarrow	Γ	\times	Bew	\times	Z
z_0		b		b		R		$z_{begin\ copy}$
$z_{begin\ copy}$		0		b		R		z_{\rightarrow}
z_{\rightarrow}		0		0		R		z_{\rightarrow}
z_{\rightarrow}		b		b		R		$z_{\rightarrow\rightarrow}$
$z_{\rightarrow\rightarrow}$		0		0		R		$z_{\rightarrow\rightarrow}$
$z_{\rightarrow\rightarrow}$		b		0		R		z_{00}
z_{00}		b		0		L		z_{\leftarrow}
z_{\leftarrow}		0		0		L		z_{\leftarrow}
z_{\leftarrow}		b		b		L		$z_{fertig?}$
$z_{fertig?}$		0		0		L		$z_{\leftarrow\leftarrow}$
$z_{\leftarrow\leftarrow}$		0		0		L		$z_{\leftarrow\leftarrow}$
$z_{\leftarrow\leftarrow}$		b		b		R		$z_{begin\ copy}$
$z_{fertig?}$		b		b		R		$z_{fertig!}$
$z_{fertig!}$		b		b		R		$z_{streiche0}$
$z_{streiche0}$		0		b		S		z_{stop}

FORMALE BESCHREIBUNG DER ARBEITSWEISE VON TURINGMASCHINEN

Nachdem wir Turingmaschinen

$$M = (\Sigma, m, T, \Gamma, Z, z_0, \delta)$$

formal spezifiziert haben, werden wir nun auch deren Arbeitsweise formal beschreiben.

Hierzu gehen wir wie folgt vor:

- (1) In *Konfigurationen* fassen wir jede mögliche aktuelle Situation von M (d.h. aktuelle Speicherbelegung und Programmzustand) zusammen.
- (2) Danach beschreiben wir Ein- und Ausgabemechanismus.
- (3) Den Programmablauf beschreiben wir durch die Angabe der *1-Schritt-Funktion*, die die einzelnen Rechenschritte von M beschreibt, d.h. einer Konfiguration von M die in einem Schritt erreichte *Nachfolgekonfiguration* zuordnet.

Hieraus lässt sich dann induktiv die gesamte *Rechnung* von M beschreiben und hiermit die *berechnete (partielle) Funktion*.

M-KONFIGURATIONEN (Idee)

Eine *M*-Konfiguration besteht aus

- (1) Bandinschrift
- (2) Position des Arbeitsfeldes
- (3) (Programm-)Zustand

(1) + (2) fassen wir als *M-Band* zusammen.

M -BÄNDER

- Wir nummerieren (*adressieren*) die Felder des Bandes mit Hilfe der ganzen Zahlen.
- Eine mögliche M -Bandinschrift ist dann eine Funktion

$$f : \mathbb{Z} \rightarrow \Gamma,$$

die der Adresse eines Feldes die Inschrift des Feldes zuordnet. Dabei ist $f(z) = b$ für fast alle z .
(D.h. nur endlich viele Felder sind nicht leer.)

BI_M bezeichnet die Menge aller M -Bandinschriften.

- Die *Position* p des Arbeitsfeldes ist dessen Adresse.
- Ein M -Band ist ein Paar (f, p) bestehend aus M -Bandinschrift f und Position p des Arbeitsfeldes.

$\text{TB}_M = \text{BI}_M \times \mathbb{Z}$ bezeichnet die Menge aller M -Bänder.

M -KONFIGURATIONEN (Definition)

Eine M -Konfiguration α ist ein Tripel

$$\alpha = (f, p, z) \in \mathbf{BI}_M \times \mathbb{Z} \times Z$$

wobei

- f die Bandinschrift
- p die Position des Arbeitsfeldes
- z der Zustand

von α sind.

Die Menge aller M -Konfigurationen wird mit \mathbf{KON}_M bezeichnet.

M -KONFIGURATIONEN (Alternative Darstellung)

Eine M -Konfiguration $\alpha = (f, p, z)$ stellen wir auch dadurch dar, dass wir den endlichen *relevanten Bandteil* von α angeben

$$\dots a_{-l} \dots a_{-1} \underbrace{a_0}_z a_1 \dots a_r \dots$$

und dabei den Zustand unter die Inschrift des Arbeitsfeldes schreiben. Dabei umfasst $a_{-l} \dots a_r$ den nichtleeren Teil der Bandinschrift.

Für späteren Gebrauch: Wählen wir die Bezeichnungen der Zustände so, dass $Z \cap \Gamma = \emptyset$, so können wir eine M -Konfiguration auch als Wort über dem Alphabet $Z \cup \Gamma$ schreiben, indem wir den Zustand hinter die Inschrift des Arbeitsfeldes setzen:

$$\dots a_{-l} \dots a_{-1} a_0 z a_1 \dots a_r \dots$$

EINGABEFUNKTION (STARTKONFIGURATIONEN)

Die *Startkonfiguration* von M bei Eingabe $\vec{x} = (x_1, \dots, x_m) \in (\Sigma^*)^m$ ist die Konfiguration

$$\alpha_M(\vec{x}) = \dots \underset{z_0}{\underline{b}} x_1 b x_2 b \dots x_m b \dots$$

Hierbei hat das Arbeitsfeld die Adresse 0 ($p = 0$).

AUSGABEFUNKTION

Die Ausgabefunktion $out_M : \text{KON}_M \rightarrow T$ ordnet jeder Konfiguration das längste direkt rechts des Arbeitsfeldes stehende Wort über T zu:

$$out_M(f, p, z) = \begin{cases} \lambda & \text{falls } f(p+1) \notin T, \\ f(p+1) \dots f(q) & \text{sonst,} \end{cases}$$

für $q > p$ minimal mit $f(q+1) \notin T$.

Die tatsächliche Ausgabe y wird der am Ende der Rechnung erreichten Konfiguration α_{stop} entnommen, d.h. $y = out_M(\alpha_{stop})$.

EINSCHRITTRELATION VON M

Die 1-Schrittrelation $\Rightarrow_M \subseteq \text{KON}_M \times \text{KON}_M$ von M ist definiert durch:

$$(f, p, z) \Rightarrow_M (\hat{f}, \hat{p}, \hat{z})$$

g.d.w. es einen Buchstaben $a \in \Gamma$ und eine Bewegung $i \in \text{Bew}$ gibt, sodass folgendes gilt:

$$\delta(z, f(p)) = (a, i, \hat{z})$$

$$\hat{f}(z) = \begin{cases} f(z) & \text{falls } z \neq p \\ a & \text{falls } z = p \end{cases}$$

$$\hat{p} = p + i$$

Hierbei fassen wir die Bewegungen als Zahlen auf: $L = -1$, $S = 0$, $R = +1$.

NB: Die 1-Schrittrelation ist eine partielle Funktion $\text{KON}_M \rightarrow \text{KON}_M$ (genauer: der Graph solch einer Funktion). Wir sprechen daher auch von der *1-Schrittfunktion*.

NACHFOLGEKONFIGURATIONEN

Gilt $\alpha \Rightarrow_M \alpha'$, so heißt α' *Nachfolgekonfiguration* von α .

NB: Jede Konfiguration besitzt höchstens eine Nachfolgekonfiguration
(\rightarrow M deterministisch!)

STOPPKONFIGURATIONEN

Besitzt α keine Nachfolgekonfiguration, so heißt α *Stoppkonfiguration*.

KONFIGURATIONENFOLGEN

Eine endliche oder unendliche Folge von Konfigurationen heißt *Konfigurationsfolge*, falls jede Konfiguration in der Folge Nachfolgekonfiguration der vorhergehenden Konfiguration ist.

Eine Konfigurationsfolge heißt *maximal*, falls sie unendlich ist oder mit einer Stoppkonfiguration endet.

Die endliche Konfigurationsfolge $\alpha_0, \dots, \alpha_n$ hat *Länge* n und wir sagen, dass α_0 *in n Schritten in α_n überführbar* bzw. α_n *aus α_0 in n Schritten erreichbar ist* ($\alpha \Rightarrow_M^n \alpha'$).

α heißt *in α' überführbar* bzw. α' *aus α erreichbar* ($\alpha \Rightarrow_M^* \alpha'$), falls es ein $n \geq 0$ gibt, sodass α in n Schritten in α' überführbar ist.

Man beachte:

- Zu jeder Konfiguration α gibt es genau eine maximale mit α beginnende Konfigurationsfolge ($\rightarrow M$ *deterministisch*).
- $\alpha \Rightarrow_M^0 \alpha' \Leftrightarrow \alpha = \alpha'$
- $\alpha \Rightarrow_M^1 \alpha' \Leftrightarrow \alpha \Rightarrow_M \alpha'$
- \Rightarrow_M^* ist der reflexive und transitive Abschluss von \Rightarrow_M

RECHNUNGEN

Die *Rechnung* von M bei Eingabe \vec{x} , $Rechnung_M(\vec{x})$, ist die (eindeutig bestimmte) maximale mit $\alpha_M(\vec{x})$ beginnende Konfigurationenfolge.

TERMINIERUNG

Ist $Rechnung_M(\vec{x})$ endlich, so *terminiert* (*hält* oder *konvergiert*) M bei Eingabe \vec{x} . Andernfalls *divergiert* M bei Eingabe \vec{x} (M *terminiert* bzw. *hält* bei Eingabe \vec{x} *nicht*).

TOTALITÄT

Terminiert M für jede Eingabe $\vec{x} \in (\Sigma^*)^m$, so ist M *total*.

DIE VON M BERECHNETE (PARTIELLE) FUNKTION

Die von $M = (\Sigma, m, \top, k, \Gamma, Z, z_0, \delta)$ berechnete partielle Funktion

$$\varphi_M : (\Sigma^*)^m \rightarrow \top^*$$

ist definiert durch:

- $\varphi_M(\vec{x})$ ist genau dann definiert, wenn die M -Rechnung bei Eingabe \vec{x} endlich ist (d.h. M bei Eingabe \vec{x} terminiert).
- Ist $\alpha_1, \dots, \alpha_l$ die M -Rechnung bei Eingabe \vec{x} , so ist

$$\varphi_M(\vec{x}) = out_M(\alpha_l).$$

NB: φ_M total $\Leftrightarrow M$ total

Nachdem wir die Arbeitsweise einer Turingmaschine M und damit die von ihr berechnete (partielle) Funktion formal definiert haben, erhalten wir die folgenden Formalisierungen der Begriffe der (partiell) berechenbaren Funktion sowie der entscheidbaren bzw. aufzählbaren Menge.

Dabei benutzen wir zur Formalisierung der Entscheidbarkeit und der Aufzählbarkeit die früher beobachteten Charakterisierungen dieser Begriffe mit Hilfe der (partiellen) Berechenbarkeit.

TURING-BERECHENBARKEIT (WORTFUNKTIONEN)

Eine (partielle) Wortfunktion $\varphi : (\Sigma^*)^m \rightarrow T^*$ ist (*partiell*) *Turing-berechenbar*, wenn es eine Turingmaschine M gibt, die φ berechnet, d.h. $\varphi = \varphi_M$.

TURING-BERECHENBARKEIT (ZAHLFUNKTIONEN)

Eine (partielle) Funktion $\varphi : \mathbb{N}^m \rightarrow \mathbb{N}$ ist (*partiell*) *Turing-berechenbar*, falls die zu ihr korrespondierende Wortfunktion $\hat{\varphi}$ bzgl. der (modifizierten) Unärdarstellung partiell Turing-berechenbar ist. Hierbei ist $Db(\hat{\varphi}) = \{(0^{n_1+1}, \dots, 0^{n_m+1}) : \varphi(n_1, \dots, n_m) \downarrow\}$ und

$$\hat{\varphi}(0^{n_1+1}, \dots, 0^{n_m+1}) = 0^{\varphi(n_1, \dots, n_m)+1}.$$

TURING-ENTSCHEIDBARKEIT

Eine m -dimensionale Sprache $L \subseteq (\Sigma^*)^m$ oder m -dimensionale Menge $A \subseteq \mathbb{N}^m$ ist *Turing-entscheidbar*, wenn ihre charakteristische Funktion Turing-berechenbar ist.

TURING-AUFZÄHLBARKEIT

Eine m -dimensionale Sprache $L \subseteq (\Sigma^*)^m$ ist *Turing-aufzählbar*, wenn sie der Definitionsbereich einer m -st. partiell Turing-berechenbaren Wortfunktion über Σ ist. Entsprechend ist eine m -dimensionale Menge $A \subseteq \mathbb{N}^m$ *Turing-aufzählbar*, wenn sie der Definitionsbereich einer m -st. partiell Turing-berechenbaren Funktion über \mathbb{N} ist.

Abschliessend betrachten wir noch die *Rechenzeit* und den *Speicherplatzbedarf* von Turingmaschinen.

Diese Konzepte werden vor allem im Teil über die Komplexität weiter betrachtet werden.

RECHENZEIT

Terminiert M bei Eingabe \vec{x} , so gilt für die *Rechenzeit* von M bei Eingabe \vec{x} :

$$time_M(\vec{x}) = \text{Länge von } Rechnung_M(\vec{x})$$

(Terminiert M bei Eingabe \vec{x} nicht, so ist $time_M(\vec{x})$ undefiniert.)

Eine totale Turingmaschine M zur Berechnung einer 1-st. Funktion heisst für eine Funktion $t : \mathbb{N} \rightarrow \mathbb{N}$ *$t(n)$ -zeitbeschränkt*, falls für fast alle Eingaben x

$$time_M(x) \leq t(|x|)$$

gilt. (D.h. t ist eine asymptotische obere Schranke für die worst-case-Zeitkomplexität von M .)

PLATZBEDARF

Um den Platzbedarf von M zu bestimmen, legen wir zunächst die Grösse $|\alpha|$ einer M -Konfiguration $\alpha = (f, p, z)$ fest:

$$|\alpha| = \max\{n \geq 0 : n = |p| \vee f(n) \neq b \vee f(-n) \neq b\}$$

Hiermit erhalten wir:

Terminiert M bei Eingabe \vec{x} und ist $\alpha_0, \dots, \alpha_n$ die Rechnung von M bei dieser Eingabe, so gilt für den *Platzbedarf* von M bei Eingabe \vec{x} :

$$space_M(\vec{x}) = \max\{|\alpha_p| : 0 \leq p \leq n\}.$$

(Terminiert M bei Eingabe \vec{x} nicht, so ist $space_M(\vec{x})$ undefiniert.)

NB: Gilt $space_M(\vec{x}) = s$, so befindet sich der relevante Bandteil (d.h. Arbeitsfeld und von b verschiedene Inschriften) während der Rechnung von M bei Eingabe \vec{x} immer innerhalb des Adressintervalls $[-s, +s]$.