
2. Turingmaschinen

Zur Formalisierung von Algorithmen benutzen wir hier Turingmaschinen. Von den vielen Varianten dieses Konzeptes, die sich in der Literatur finden, greifen wir das Konzept der on-line k -Band-Turingmaschinen mit Programmen aus bedingten Anweisungen heraus. Wir werden dieses Konzept zunächst intuitiv, dann formal beschreiben, und schließlich aus ihm die Grundbegriffe der Berechenbarkeitstheorie ableiten. Bei den anderen Turingmaschinen-Varianten, die wir in der Vorlesung benutzen werden, begnügen wir uns mit der intuitiven Beschreibung.

Allgemein ist eine Turingmaschine M eine Rechenmaschine, die in *Schritten* arbeitet. Die durchlaufene Schrittfolge, genannt *Rechnung*, hängt dabei i.a. von der Eingabe ab, muss aber durch diese nicht eindeutig festgelegt sein. D.h. die Maschine kann *nicht-deterministisch* (*nd.*) sein. Ist die Rechnung durch die Eingabe eindeutig festgelegt, so spricht man von einer *deterministischen* (*det.*) Maschine. Ziel der Rechnung kann die Überführung der Eingabe in eine Ausgabe sein (d.h. M berechnet eine Funktion; man nennt M dann einen *Transduktor*) oder die Feststellung, ob die Eingabe eine vorgegebene Eigenschaft hat (d.h. M löst ein Problem oder erkennt eine Sprache; man nennt M dann einen *Akzeptor*).

Bei den Funktionseinheiten einer Rechenmaschine unterscheidet man zwischen dem *Speicher* und der *Kontrolle* oder *Steuerung*, letztere bestehend aus *Programm* und *Zeiger* auf die nächste auszuführende Instruktion (*Programmzustand*).

Bei einer Turingmaschine besteht der Speicher aus einer festen Anzahl von (*Turing*-) *Bändern*. Diese sind in *Felder* eingeteilt und nach beiden Seiten unbeschränkt. Jedes Feld kann einen Buchstaben aus einem vorgegebenen *Bandalphabet* aufnehmen. Zu jedem Zeitpunkt der Rechnung sind jedoch nur endlich viele Felder belegt, d.h. der Speicher(inhalt) ist stets endlich. (Aus notationellen Gründen werden wir in der formalen Definition leere Felder mit dem *Leerzeichen* (*blank*) beschriften.)

Zu jedem Band gibt es einen *Lese-Schreibkopf*, der auf ein Feld (das *Arbeitsfeld*) zeigt, und dieses lesen und (anschließend) überschreiben kann. Weiter kann in jedem Schritt jeder Kopf um ein Feld nach links bzw. nach rechts bewegt werden.

Die Kontrolle der Maschine befindet sich zu Beginn jedes Rechenschrittes in einem von endlich vielen möglichen *Zuständen*, wobei ein Zustand als *Startzustand*, in dem jede Rechnung beginnt, ausgezeichnet ist. Das *Programm* bestimmt die in den einzelnen Schritten auszuführenden Speicheroperationen und den nächsten Zustand (*Nachfolgezustand*) der Kontrolle. Dabei werden in Abhängigkeit des alten Zustands und der Inschriften der Arbeitsfelder deren neue Inschriften (Druckbefehle) und Positionen (Bewegungsbefehle) sowie der neue Zustand festgelegt. Alternativ kann das Programm in bestimmten Situationen das Stoppen der Maschine veranlassen.

Die Eingabe wird auf das erste Band rechts des Arbeitsfeldes in den ansonsten leeren Speicher geschrieben. Bei mehreren Eingaben werden diese durch Leerfelder getrennt. Die Ausgabe (bei einem Transduktor) wird am Ende der Rechnung dem letzten Band entnommen und befindet sich dort wieder rechts vom Arbeitsfeld.

2.1 DEFINITION. a) Sei $k \geq 1$ und seien Σ, T Alphabete. Eine *nichtdeterministische (nd.) k -Band-Turingmaschine M zur Berechnung einer m -stelligen Funktion $f : (\Sigma^*)^m \rightarrow T^*$* ist ein 8-Tupel

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

mit folgenden Komponenten:

- Σ ist das *Eingabealphabet*,
- m ist die *Stelligkeit* der zu berechnenden Funktion,
- T ist das *Ausgabealphabet*,
- k ist die *Bandanzahl*,
- Γ ist ein Alphabet mit $\Sigma \cup T \subseteq \Gamma$ 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 Relation auf $Z \times \Gamma^k \times \Gamma^k \times \text{Bew}^k \times Z$, genannt das *Programm*, wobei $\text{Bew} = \{-1, 0, +1\}$ die Menge der *Bewegungen* ist. (Hierbei stehen $-1, 0, +1$ für Linksbewegung, Stehenbleiben und Rechtsbewegung und beschreiben die von diesen Kopfbewegungen bewirkten *Adressänderungen* (siehe unten) des Arbeitsfeldes.)

b) Eine *nd. k -Band-Turingmaschine M zur Erkennung einer m -stelligen Sprache $L \subseteq (\Sigma^*)^m$* ist ein 8-Tupel

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

wobei $\Sigma, m, k, \Gamma, Z, z_0, \delta$ wie oben definiert sind und E eine Teilmenge von Z ist, genannt die Menge der *Endzustände* von M .

c) Die Maschine M ist *deterministisch*, falls für $i \in \{0, 1\}$ und

$$\vec{v}^i = (z^i, a_1^i, \dots, a_k^i, \hat{a}_1^i, \dots, \hat{a}_k^i, j_1^i, \dots, j_k^i, \hat{z}^i) \in \delta$$

mit $\vec{v}^0 \neq \vec{v}^1$ gilt, dass $(z^0, a_1^0, \dots, a_k^0) \neq (z^1, a_1^1, \dots, a_k^1)$.

Statt Turingmaschine schreiben wir häufig kurz TM. Das Programm δ einer nd. TM M gibt man meist in Tabellenform oder als Liste der Elemente an. Ein Element \vec{v} von δ nennt man entsprechend eine *Programmzeile* oder eine *Instruktion* von δ (oder M). Für eine Instruktion

$$(z, a_1, \dots, a_k, \hat{a}_1, \dots, \hat{a}_k, j_1, \dots, j_k, \hat{z})$$

nennen wir (z, a_1, \dots, a_k) den *Bedingungs-* und $(\hat{a}_1, \dots, \hat{a}_k, j_1, \dots, j_k, \hat{z})$ den *Operationsteil*, da ersterer festlegt, ob die Instruktion in einer Situation anwendbar ist und letzterer die Wirkung der Instruktion beschreibt. Für deterministisches M legt der Bedingungssteil einer Instruktion diese eindeutig fest. Hier schreibt man daher das Programm δ häufig auch als partielle Funktion $\delta : Z \times \Gamma^k \rightarrow \Gamma^k \times \text{Bew}^k \times Z$.

Zur formalen Beschreibung der Arbeitsweise der Turingmaschine M definiert man die möglichen Zustände, in denen sich M befinden kann, als Konfigurationen von M und bestimmt mit der 1-Schrittrelation die vom Programm δ in einem Schritt erlaubten Konfigurationsübergänge. Die Rechnung ergibt sich dann hieraus durch Iteration.

Zur Beschreibung dieser Konzepte nummerieren wir die Felder eines jeden Turingbandes (mit den ganzen Zahlen) durch, wobei das Arbeitsfeld zu Beginn der Rechnung die Nummer 0 erhält. Diese Ordnungszahl eines Feldes nennen wir auch dessen *Adresse*. Man beachte, dass eine elementare Links- bzw. Rechtsbewegung eines Lese-Schreibkopfes gerade dem De- bzw. Inkrementieren der Adresse des zugehörigen Arbeitsfeldes entspricht. (Diese Entsprechung haben wir bei der oben gewählten Notation der Bewegungsbefehle bereits berücksichtigt.) Man beachte auch, dass die Adressierung nur zur Beschreibung der Arbeitsweise der Turingmaschine dient, von dieser jedoch nicht verwendet wird.

2.2 DEFINITION. Sei M eine k -Band-TM wie in Definition 2.1.

- Eine (M -)Bandinschrift ist eine Funktion $f : \mathbb{Z} \rightarrow \Gamma$, wobei $f(z) = b$ für fast alle $z \in \mathbb{Z}$. Ein (M -)Band ist ein Paar (f, p) , wobei f eine Bandinschrift und p eine ganze Zahl, die *Position des Arbeitsfeldes*, ist. Die Menge der (M -)Bänder wird mit TB_M bezeichnet.
- Eine (M -)Konfiguration ist ein $(k+1)$ -Tupel $(z, \beta_1, \dots, \beta_k) \in Z \times TB_M^k$ bestehend aus einem M -Zustand und k M -Bändern. Mit $KON_M = Z \times TB_M^k$ bezeichnen wir die Menge der M -Konfigurationen.
- Die *Startkonfiguration* von M bei Eingabe $\vec{w} = (w_1, \dots, w_m) \in (\Sigma^*)^m$ ist die Konfiguration

$$\alpha_M(\vec{w}) = (z_0, (f_{\vec{w}}, 0), (f_{\lambda}, 0), \dots, (f_{\lambda}, 0))$$

wobei

$$f_{\vec{w}}(n_l + s) = w_l(s)$$

für $1 \leq l \leq m$, $n_l = |w_1| + \dots + |w_{l-1}| + l$ (also $n_1 = 1$) und $0 \leq s < |w_l|$ und

$$f_{\vec{w}}(z) = b$$

sonst; und $f_{\lambda}(z) = b$ für alle $z \in \mathbb{Z}$.

Konfigurationen schreiben wir meist weniger formal, indem wir für jedes Band den relevanten Bandteil, der die beschrifteten Felder und das Arbeitsfeld enthält, angeben. Hierbei unterstreichen wir die Inschrift des Arbeitsfeldes und schreiben den Zustand unter das Arbeitsfeld des ersten Bandes. Die Startkonfiguration $\alpha_M(w_1, \dots, w_m)$ lässt sich dann folgendermaßen beschreiben:

$$\begin{array}{c} \underline{b} \ w_1 b w_2 \dots b w_m b \\ z_0 \\ \underline{b} \\ \vdots \\ \underline{b} \end{array}$$

2.3 DEFINITION. Sei M eine k -Band-TM wie in Definition 2.1.

- Die 1-Schrittrelation $\Rightarrow_M \subseteq KON_M \times KON_M$ von M ist definiert durch:

$$(z, (f_1, p_1), \dots, (f_k, p_k)) \Rightarrow_M (\hat{z}, (\hat{f}_1, \hat{p}_1), \dots, (\hat{f}_k, \hat{p}_k))$$

g.d.w. es Buchstaben $a_1, \dots, a_k \in \Gamma$ und Bewegungen $i_1, \dots, i_k \in \text{Bew}$ gibt, sodass folgendes gilt:

$$(z, f_1(p_1), \dots, f_k(p_k), a_1, \dots, a_k, i_1, \dots, i_k, \hat{z}) \in \delta,$$

$$\hat{f}_l(z) = \begin{cases} f_l(z) & \text{falls } z \neq p_l \\ a_l & \text{falls } z = p_l \end{cases}$$

$$\hat{p}_l = p_l + i_l$$

wobei $1 \leq l \leq k$.

- b) Sind α und β M -Konfigurationen, so heisst β *Nachfolgekonfiguration* von α , falls $\alpha \Rightarrow_M \beta$ gilt. Besitzt α keine Nachfolgekonfiguration, so ist α eine *Stoppkonfiguration*. Ist der Zustand von α ein Endzustand, so ist α eine *Endkonfiguration*. (Ist M ein Transduktor, so ist keine M -Konfiguration eine Endkonfiguration.)
- c) Die *Mehrschrittrelation* $\overset{*}{\Rightarrow}_M$ von M ist der reflexive und transitive Abschluss von \Rightarrow_M . Gilt $\alpha \overset{*}{\Rightarrow}_M \beta$, so heisst β aus α *erreichbar* oder α in β *überföhrbar*. Ist β aus der Startkonfiguration $\alpha_M(\vec{w})$ erreichbar, so heisst β (*bei Eingabe \vec{w}*) *erreichbar*.
- d) Eine *M -Konfigurationenfolge* ist eine endliche oder unendliche Folge $(\alpha_1, \dots, \alpha_n$ bzw. $\alpha_1, \alpha_2, \dots)$ von M -Konfigurationen, in der (mit Ausnahme der ersten Konfiguration) jede Konfiguration Nachfolgekonfiguration der vorhergehenden Konfiguration ist. Die *Länge* der Konfigurationenfolge $\alpha_1, \dots, \alpha_n$ ist $n - 1$, und wir sagen α_n ist aus α_1 *in $n - 1$ Schritten erreichbar* und schreiben hierfür $\alpha_1 \overset{n-1}{\Rightarrow}_M \alpha_n$.
- e) Eine *unendliche M -Rechnung* bei Eingabe $\vec{w} = (w_1, \dots, w_m)$ ist eine unendliche M -Konfigurationenfolge, die mit $\alpha_M(\vec{w})$ beginnt und die keine Endkonfiguration enthält. Eine *endliche M -Rechnung* bei Eingabe \vec{w} ist eine endliche, mit $\alpha_M(\vec{w})$ beginnende M -Konfigurationenfolge $\alpha_1, \dots, \alpha_l$, deren letztes Glied α_l eine Stopp- oder Endkonfiguration ist und die bis auf möglicherweise α_l keine Endkonfiguration enthält. Ist α_l Endkonfiguration, so heisst die Rechnung *akzeptierend*, sonst *verwerfend*. Die *Länge* einer Rechnung ist die Länge der zugehörigen Konfigurationenfolge. M *terminiert* oder *konvergiert* bei Eingabe \vec{w} , falls es bei dieser Eingabe eine akzeptierende M -Rechnung gibt oder alle M -Rechnungen endlich sind; andernfalls *divergiert* M bei Eingabe \vec{w} .
- f) Der *Rechenbaum* $\text{RB}_M(\vec{w})$ von M bei Eingabe \vec{w} ist der Baum, dessen Knoten mit M -Konfigurationen wie folgt markiert sind. Die Wurzel ist mit der Startkonfiguration $\alpha_M(\vec{w})$ markiert. Ist ein Knoten mit α markiert, so hat dieser Knoten so viele Söhne, wie α Nachfolgekonfigurationen besitzt, und diese sind mit den Nachfolgekonfigurationen von α markiert.
- g) M heisst *total*, wenn M bei jeder Eingabe konvergiert.

Ist die zugrundegelegte TM M aus dem Kontext klar, so lassen wir in den oben eingeföhrten Notationen den Verweis auf M weg. Man beachte, dass $\alpha \overset{0}{\Rightarrow} \beta$ genau dann gilt, wenn $\alpha = \beta$, $\alpha \overset{1}{\Rightarrow} \beta$ g.d.w. $\alpha \Rightarrow \beta$, und $\alpha \overset{*}{\Rightarrow} \beta$ g.d.w. $\alpha \overset{n}{\Rightarrow} \beta$ für ein $n \geq 0$.

Weiter beobachtet man, dass die Knoten der Tiefe n im Rechenbaum $\text{RB}_M(\vec{w})$ mit den Konfigurationen α markiert sind, die aus $\alpha_M(\vec{w})$ in n Schritten erreichbar sind, und

dass daher (bei Identifizierung des Rechenbaumes mit den als Markierungen vorkommenden Konfigurationen)

$$\text{RB}_M(\vec{w}) = \{\alpha : \alpha_M(\vec{w}) \xrightarrow{*} \alpha\}$$

gilt. Die Akzeptanz können wir daher auch mit Hilfe von $\xrightarrow{*}$ durch

$$\vec{w} \in L(M) \Leftrightarrow \exists \alpha (\alpha_M(\vec{w}) \xrightarrow{*} \alpha \ \& \ \alpha \text{ ist Endkonfiguration}) \quad (2.1)$$

ausdrücken, d.h. M akzeptiert die Eingabe \vec{w} , wenn bei dieser Eingabe eine Endkonfiguration erreichbar ist.

Weiter beachte man, dass die unendlichen bzw. endlichen M -Rechnungen bei einer Eingabe \vec{w} gerade die von der Wurzel ausgehenden unendlichen bzw. an einem Blatt endenden Pfade im Rechenbaum $\text{RB}_M(\vec{w})$ sind. Ist M deterministisch, so besitzt jede Konfiguration höchstens eine Nachfolgekonfiguration, weshalb es zu jeder Eingabe genau eine M -Rechnung gibt (und M konvergiert entsprechend genau dann, wenn diese Rechnung endlich ist). Hier degenerieren die Rechenbäume $\text{RB}_M(\vec{w})$ also zu jeweils einem Pfad, d.h. haben Verzweigungsgrad 1. Für nichtdeterministisches M kann man dem Programm δ eine obere Schranke $s \in \mathbb{N}$ für die Anzahl der Nachfolgekonfigurationen einer Konfiguration entnehmen, woraus man eine uniforme Schranke für den Verzweigungsgrad der Rechenbäume von M erhält. (Man kann hierbei s als die maximale Anzahl von Instruktionen in δ mit identischem Bedingungsteil wählen.) Nach Königs Lemma¹ sind daher Rechenbäume, die nur endliche Rechnungen enthalten, ebenfalls endlich.

Im Falle von Transduktoren definieren wir die berechnete Funktion nur für den deterministischen Fall. Für $M = (\Sigma, m, T, k, \Gamma, Z, z_0, \delta)$ ist dies eine partielle Funktion $f : (\Sigma^*)^m \rightarrow T^*$, wobei $f(\vec{w})$ genau dann definiert ist, wenn die Rechnung von M bei Eingabe \vec{w} endlich ist. In diesem Fall ist $f(\vec{w})$ das längste Wort aus T^* , das am Ende der Rechnung auf dem letzten (= k -ten) Band rechts des Arbeitsfeldes steht.

2.4 DEFINITION. a) Die von der det. TM $M = (\Sigma, m, T, k, \Gamma, Z, z_0, \delta)$ berechnete partielle Funktion $\text{res}_M : (\Sigma^*)^k \rightarrow T^*$ ist definiert durch:

- (i) $\text{res}_M(\vec{w})$ ist genau dann definiert, wenn die M -Rechnung bei Eingabe \vec{w} endlich ist.
- (ii) Ist $\alpha_1, \dots, \alpha_k$ die M -Rechnung bei Eingabe \vec{w} , ist

$$\alpha_i = (z, (f_1, p_1), \dots, (f_k, p_k))$$

und ist q die kleinste Zahl $> p_k$ mit $f_k(q) \notin T$, so ist

$$\text{res}_M(\vec{w}) = f_k(p_k + 1) \dots f_k(q - 1).$$

(Ist $q - 1 < p_k + 1$, so bedeutet dies, dass $\text{res}_M(\vec{w}) = \lambda$.)

b) Eine (partielle) Funktion $\varphi : (\Sigma^*)^k \rightarrow T^*$ ist (*partiell*) *Turing-berechenbar* oder (*partiell*) *rekursiv*, wenn sie von einer deterministischen Turingmaschine berechnet wird.

¹Königs Lemma besagt, dass jeder endlich (nicht notwendigerweise beschränkt) verzweigende unendliche Baum B einen unendlichen Pfad w besitzt. Die Anfangsstücke $w \upharpoonright n = w(0) \dots w(n)$ von w definiert man induktiv, wobei man (mit Hilfe des Auswahlaxioms) sicherstellt, dass der Teilbaum unterhalb von $w \upharpoonright n$ unendlich ist.

Man beachte, dass res_M genau dann total ist, wenn M total ist. Nach Churchscher These sind die (partiellen) Turing-berechenbaren Funktionen genau die (partiellen) Funktionen, die durch einen Algorithmus berechnet werden können. Man beachte, dass bei einer totalen Turing-berechenbaren Funktion die Maschine bei jeder Eingabe den Funktionswert nach endlich vielen Schritten ausgibt. Bei einer partiellen Turing-berechenbaren Funktion kann die Rechnung jedoch bei Eingaben, für die die Funktion nicht definiert ist, unendlich sein.

Bei der Erkennung von Sprachen lassen wir auch nichtdeterministische Maschinen zu, indem wir festlegen, dass eine Eingabe zu der erkannten Sprache genau dann gehört, wenn es zumindest eine Rechnung gibt, die die Eingabe akzeptiert.

2.5 DEFINITION. a) Die von der Turingmaschine $M = (\Sigma, m, k, \Gamma, Z, z_0, E, \delta)$ erkannte (m -dimensionale) Sprache $L(M) \subseteq (\Sigma^*)^m$ ist die Sprache

$$L(M) = \{\vec{w} \in (\Sigma^*)^m : M \text{ akzeptiert } \vec{w}\}.$$

Hierbei *akzeptiert* M die Eingabe \vec{w} , falls es eine akzeptierende Rechnung von M bei Eingabe \vec{w} gibt.

b) Eine Sprache L ist *Turing-aufzählbar* oder *rekursiv aufzählbar (r.a.)*, falls L von einer deterministischen Turingmaschine erkannt wird. L ist *Turing-entscheidbar* oder *rekursiv*, wenn L von einer totalen deterministischen Turingmaschine erkannt wird.

Man beachte die i. a. bestehende Asymmetrie in der Definition des Akzeptierens und Verwerfens einer Eingabe \vec{w} durch die Maschine M . Lediglich für totales deterministisches M besteht hier eine Symmetrie, da es hier eine durch die Eingabe eindeutig festgelegte endliche Rechnung gibt und die letzte Konfiguration dieser Rechnung das Akzeptanzverhalten bestimmt (nämlich bei einer Endkonfiguration akzeptiert, bei einer Nicht-Endkonfiguration verworfen wird). Ist M deterministisch aber nicht total, so erfordert die Akzeptanz wieder eine endliche (mit einer Endkonfiguration endende) Rechnung, während das Verwerfen auf zwei Arten realisiert werden kann, nämlich durch eine endliche (mit einer Nicht-Endkonfiguration endende) Rechnung oder durch eine *unendliche* Rechnung. Bei nichtdeterministischem M kann bei Eingabe \vec{w} ein Teil der Rechnungen akzeptieren, ein Teil verwerfen. Hier dominieren die akzeptierenden Rechnungen, d. h. die Eingabe wird akzeptiert, wenn wenigstens eine Rechnung akzeptierend ist. Die Eingabe wird also nur verworfen, wenn alle Rechnungen verwerfend oder unendlich sind. Bei der Definition der Totalität nichtdeterministischer Maschinen haben wir dieses Akzeptanzkriterium berücksichtigt: Totalität garantiert, dass bei jeder Eingabe das Akzeptanzverhalten nach endlich vielen Schritten feststeht, obwohl unendliche Rechnungen zugelassen werden.

Die Unterschiede beim Akzeptieren durch totale und nichttotale deterministische Turingmaschinen sind verantwortlich für den wesentlichen Unterschied zwischen einer rekursiven und einer rekursiv aufzählbaren Sprache L : Im Falle der Rekursivität lässt sich für jede Eingabe w nach endlich vielen Schritten die Frage beantworten, ob das Wort w zu der Sprache L gehört oder nicht, während im Falle der rekursiven Aufzählbarkeit solch eine Antwort nur im positiven Fall (d.h. für $w \in L$) nach endlich vielen Schritten erfolgen muss, im negativen Fall die Maschine aber unendlich lange laufen darf ohne eine Antwort zu geben.

Nach Churchscher These sind die rekursiven bzw. rekursiv aufzählbaren Sprachen gerade die effektiv entscheidbaren bzw. aufzählbaren Sprachen (siehe "Einführung in

die Theoretische Informatik"). Aus Bequemlichkeit werden wir häufiger auf diese These zurückgreifen und die Rekursivität oder rekursive Aufzählbarkeit von Sprachen nur anschaulich nachweisen.

In der Definition der rekursiven und r.a. Sprachen hätten wir auch nichtdeterministische Turingmaschinen verwenden können. Um dies zu zeigen, definieren wir zunächst:

2.6 DEFINITION. Zwei Turingmaschinen M und \hat{M} sind *äquivalent*, falls sie dieselbe partielle Funktion berechnen (im Falle von Transduktoren) bzw. dieselbe Sprache erkennen (im Falle von Akzeptoren).

2.7 LEMMA. Zu jedem (totalen) nichtdeterministischen Turingakzeptor M gibt es einen äquivalenten (totalen) deterministischen Turingakzeptor \hat{M} .

BEWEISIDEE. Bei Eingabe \vec{w} simuliert \hat{M} die Maschine M , indem es den Rechenbaum $RB_M(\vec{w})$ der Breite nach durchsucht und genau dann akzeptiert, wenn es eine Endkonfiguration findet. Ist M total, so ist die Maschine \hat{M} ebenfalls total: Akzeptiert M die Eingabe \vec{w} , so besitzt M bei dieser Eingabe eine akzeptierende Rechnung, weshalb \hat{M} nach endlich vielen Schritten die zugehörige Endkonfiguration findet und ebenfalls akzeptiert; akzeptiert M die Eingabe nicht, so ist nach Definition der Totalität jede M -Rechnung bei dieser Eingabe und damit auch der Rechenbaum insgesamt endlich, weshalb \hat{M} die Simulation nach endlich vielen Schritten abschließt und verwirft.

Die Rekursivität von zahlentheoretischen Funktionen und Mengen definieren wir über deren Binärcodierung.

2.8 DEFINITION. Eine (partielle) Funktion $f : \mathbb{N}^n \rightarrow \mathbb{N}$ ist (partiell) *rekursiv*, falls die Funktion $\hat{f} : (\Sigma_2^*)^n \rightarrow \Sigma_2^*$ mit $\hat{f}(\underline{m}_1, \dots, \underline{m}_n) = \underline{f(m_1, \dots, m_n)}$ (partiell) rekursiv ist. Eine Menge $A \subseteq \mathbb{N}^n$ ist *rekursiv (rekursiv aufzählbar)*, falls die n -dimensionale Sprache $\hat{A} = \{(\underline{m}_1, \dots, \underline{m}_n) : (m_1, \dots, m_n) \in A\}$ rekursiv (rekursiv aufzählbar) ist.

Mit $\text{PFREK}_n^{(m)}$, $\text{FREK}_n^{(m)}$, $\text{REK}_n^{(m)}$, $\text{RA}_n^{(m)}$ bezeichnen wir die Klassen der partiell rekursiven Funktionen bzw. total rekursiven Funktionen $\varphi : (\Sigma_n^*)^m \rightarrow \Sigma_n^*$ und der rekursiven bzw. r.a. Teilmengen von $(\Sigma_n^*)^m$. Für 1-stellige Funktionen bzw. Mengen über dem binären Alphabet lassen wir die Parameter $m = 1$ und $n = 2$ weg. Nach Definition 2.8 werden letztere Funktionen und Mengen weiterhin mit denen über \mathbb{N} identifiziert. Die wichtigsten Eigenschaften der (partiell) rekursiven Funktionen und der rekursiven und r.a. Mengen, die wir benötigen werden, werden wir in Abschnitt 4 zusammenfassen.

Neben der oben definierten on-line- k -Band-TM benutzen wir folgende Varianten (die Formalisierung überlassen wir als Übung):

1. *off-line-Turingmaschinen:* Hier werden die k Arbeitsbänder durch ein Eingabe- und ein Ausgabeband ergänzt. Auf das Eingabeband darf nur lesend zugegriffen werden, wobei der Lesekopf uneingeschränkt bewegt werden darf. Auf das Ausgabeband darf nur geschrieben werden, und zwar nur Buchstaben aus dem Ausgabealphabet, und der Schreibkopf darf nicht nach links zurückgesetzt werden. Zu Beginn der Rechnung wird die Eingabe auf das Eingabeband geschrieben. Das Ergebnis ist die Inschrift des Ausgabebandes am Ende der Rechnung. (Bei einem Akzeptor fehlt natürlich das Ausgabeband.)

2. *Halbband-Turingmaschinen*: Hier sind Turingbänder unbeschränkt nur nach rechts hin. Die Adressierung der Felder erfolgt also durch die natürlichen Zahlen. Verlangt das Programm bei Arbeitsfeld 0 eine Linksbewegung, so wird die zugehörige Instruktion nicht ausgeführt und es kommt zu einem (Fehler-)Stopp (mit Verwerfen der Eingabe im Falle eines Akzeptors).

Der *Typ* einer Turingmaschine wird durch den Ein-Ausgabemechanismus (on-line vs. off-line), das Bandformat (Vollband vs. Halbband), und den Arbeitsmodus (det. vs. nd.) festgelegt. Im folgenden wird eine Turingmaschine immer eine deterministische on-line-Maschine sein, wenn nicht ausdrücklich anders gesagt.