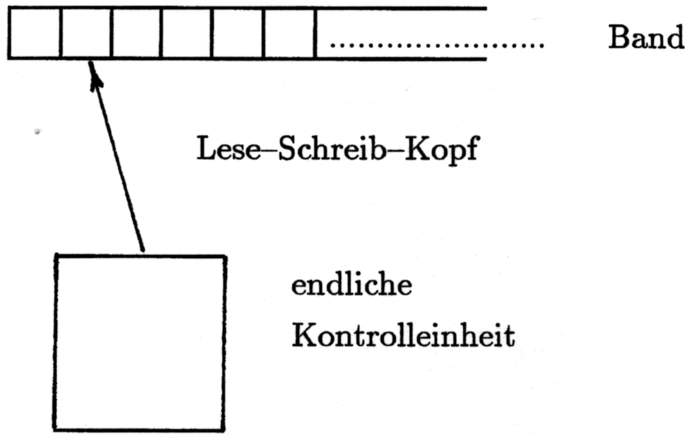
## 1.4. Turing–Maschinen und rekursiv aufzählbare Sprachen

Um zu beweisen, daß eine bestimmte Funktion "berechenbar" ist, oder ein gewisses Minimum an Zeit zur Berechnung benötigt (untere Komplexitätsschranke), brauchen wir ein Maschinenmodell, das zwar so allgemein aber einfacher als eine RAM oder RASP ist. Dieses Modell ist die Turing–Maschine.

**Definition 1.4.1:** Eine *Turing–Maschine* $M$ können wir uns folgendermaßen vorstellen:




Sie besteht aus einer *endlichen Kontrolleinheit*, welche im Inneren eine endliche Anzahl von Zuständen aus einer *Zustandsmenge* $Q = \{q_0, \ldots, q_n\}$ annehmen kann, einem in eine Richtung unendlichen *Band*, welches in Zellen eingeteilt ist, deren jede ein Symbol aus einem endlichen *Alphabet von Bandsymbolen* $\Gamma = \{\gamma_0, \ldots, \gamma_m\}$ speichern kann — $\gamma_0$ wird dabei als das Blank Symbol $\sqcup$ aufgefaßt und eine Untermenge $\Sigma$ von $\Gamma \setminus \{\sqcup\}$ ist das *Eingabealphabet* — , sowie aus einem *Lese–Schreib–Kopf*, welcher immer auf eine Bandzelle zeigt.

Wie funktioniert nun die Turing–Maschine $M$? Wenn die Maschine zu arbeiten beginnt, ist die endliche Kontrolleinheit im Anfangszustand $q_0$, und bis auf endlich viele Zellen am linken Ende des Bandes sind alle Bandzellen mit $\sqcup$'s belegt. In diskreten Zeitschritten, abhängig vom Zustand und vom Bandsymbol, auf welches der LS–Kopf gerade zeigt (Symbol in der *Arbeitszelle*), geschieht folgendes:

- $M$ nimmt einen neuen Zustand an,
- schreibt ein Bandsymbol in die Arbeitszelle, und
- bewegt den LS–Kopf um eine Position nach rechts (R) oder nach links (L) oder läßt ihn stationär (S).

Erreicht dabei $M$ einen Zustand aus einer Menge von *Endzuständen* $F$ (Teilmenge der Zustandsmenge), so terminiert die Berechnung erfolgreich, $M$ *akzeptiert* die Eingabe. Ansonsten fährt $M$ in der Berechnung so lange fort, als zum jeweiligen Zustand und Symbol in der Arbeitszelle ein Nachfolgerzustand erklärt ist. Eine Berechnung, welche nicht erfolgreich terminiert, kann also auch unendliche Länge haben. ■

**Beispiel 1.4.1:** Als Beispiel betrachten wir die Turing–Maschine $M_1$, welche erkennt, ob eine endliche Folge $x$ über $\{0,1\}$ die Form $0^n1^n$ hat, für eine natürliche Zahl $n$. Bei Anfang der Berechnung steht die Folge $x$ in den ersten Zellen des Bandes von $M_1$, gefolgt von einer unendlichen Folge von $\sqcup$'s. $M_1$ führt folgende Berechnungsschleife aus:

> $M_1$ ersetzt die am weitesten links stehende 0 durch $X$, bewegt den LS–Kopf nach rechts bis zur am weitesten links stehenden 1, ersetzt diese durch $Y$, bewegt den LS–Kopf nach links bis zum ersten $X$ und anschließend um eine Position nach rechts zur am weitesten links stehenden 0 und wiederholt die Schleife.

Findet nun $M_1$ bei der Suche nach einer 1 anstatt dessen ein $\sqcup$, so stoppt $M_1$ die Berechnung und gibt "nein" als Antwort. Falls $M_1$, nachdem es eine 1 durch $Y$ ersetzt hat, keine weitere 0 mehr findet, so prüft es, ob noch 1en vorhanden sind, und gibt in diesem Fall die Antwort "nein", andernfalls die Antwort "ja".

Etwas formaler könnten wir $M_1$ wie folgt angeben:

Zustandsmenge .... $Q = \{q_0, q_1, q_2, q_3, q_4\}$,
Eingabealphabet ... $\Sigma = \{0, 1\}$,
Bandalphabet ..... $\Gamma = \{\sqcup, 0, 1, X, Y\}$,
Endzustände ...... $F = \{q_4\}$,

Überführungsfunktion $\delta$:

| Zustand | Symbol | | | | |
|---|---|---|---|---|---|
| | $\sqcup$ | 0 | 1 | $X$ | $Y$ |
| $q_0$ | – | $(q_1, X, R)$ | – | – | $(q_3, Y, R)$ |
| $q_1$ | – | $(q_1, 0, R)$ | $(q_2, Y, L)$ | – | $(q_1, Y, R)$ |
| $q_2$ | – | $(q_2, 0, L)$ | – | $(q_0, X, R)$ | $(q_2, Y, L)$ |
| $q_3$ | $(q_4, \sqcup, R)$ | – | – | – | $(q_3, Y, R)$ |
| $q_4$ | – | – | – | – | – |

Jeden Zustand könnte man sich vorstellen als eine Anweisung oder eine Gruppe von Anweisungen in einem Programm.

– Der Zustand $q_0$ wird am Start angenommen und auch jeweils unmittelbar vor der Ersetzung der am weitesten links stehenden 0 durch $X$. Findet $M_1$ eine 0 vor, so ersetzt es diese durch $X$, geht über in $q_1$, und bewegt den LS–Kopf eine Position nach rechts. Findet $q_0$ ein $Y$ vor, so nimmt $M_1$ den Zustand $q_3$ an.

– $q_1$ wird dazu benutzt, um von links nach rechts nach der ersten 1 zu suchen. Findet $M_1$ eine 1, so wird sie durch $Y$ ersetzt, und $M_1$ nimmt den neuen Zustand $q_2$ an. Wird vor einer 1 ein $\sqcup$ oder ein $X$ gefunden, so bricht die Berechnung ab, und die Eingabe wird zurückgewiesen.

– $q_2$ wird benutzt um von rechts nach links nach dem ersten $X$ zu suchen. Wird ein $X$ gefunden, so nimmt $M_1$ den Zustand $q_0$ an und bewegt den LS–Kopf eine Position nach rechts.

– Im Zustand $q_3$ wird der LS–Kopf über $Y$'s hinweg nach rechts bewegt um zu prüfen, ob noch 1en auf dem Band vorkommen. Ist das erste Symbol nach den $Y$'s ein $\sqcup$, so nimmt $M_1$ den Zustand $q_4$ an und die Eingabefolge wird damit akzeptiert. Andernfalls wird die Eingabefolge zurückgewiesen. ■

**Definition 1.4.1:** (Fortsetzung) Etwas formaler wird eine *Turing–Maschine* (kurz *TM*) so angegeben wie im obigen Beispiel, nämlich als ein 6–tupel $(Q, \Sigma, \Gamma, q_0, F, \delta)$, wobei $Q$ die Zustandsmenge ist, $\Sigma$ das Eingabealphabet, $\Gamma$ das Bandalphabet, $q_0$ der Startzustand, $F$ die Menge der Endzustände und $\delta$ die Überführungsfunktion. $\delta$ ist eine partielle Funktion von $Q \times \Gamma$ in $Q \times \Gamma \times \{L, R, S\}$. ■

**Definition 1.4.2:** Sei $M$ ein TM. Eine *Konfiguration* von $M$ legt den Zustand von $M$ zu einem bestimmten Zeitpunkt fest, also eine Folge von Bandsymbolen $\alpha_1 \dots \alpha_k$, welche links vom LS–Kopf auf dem Band stehen, eine Folge von Bandsymbolen $\alpha_{k+1} \dots \alpha_l$, welche rechts vom LS–Kopf auf dem Band stehen einschließlich dem Symbol in der Arbeitszelle (die unendlich vielen $\sqcup$'s am rechten Bandende werden dabei vernachlässigt), sowie den Zustand $q$ der endlichen Kontrolleinheit. Alle anderen Bandsymbole rechts von der Position $l$ sind $\sqcup$. Wir schreiben dafür

$$\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_l.$$

Also $\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_l$ und $\alpha_1 \dots \alpha_k q \alpha_{k+1} \dots \alpha_l \sqcup$ sind nur zwei verschiedene Notationen für ein und dieselbe Konfiguration.

Eine Konfiguration $\beta_1 \ldots \beta_k q \beta_{k+1} \ldots \beta_m$ geht aus $\alpha_1 \ldots \alpha_l p \alpha_{l+1} \ldots \alpha_m$ hervor,

$$\alpha_1 \ldots \alpha_l p \alpha_{l+1} \ldots \alpha_m \quad \vdash \quad \beta_1 \ldots \beta_k q \beta_{k+1} \ldots \beta_m,$$

wenn $\alpha_i = \beta_i$ für $i \in \{1, \ldots, l, l+2, \ldots, m\}$ und

$$\delta(p, \alpha_{l+1}) = (q, \beta_{k+1}, S) \quad \text{und} \quad k = l,$$

oder

$$\delta(p, \alpha_{l+1}) = (q, \beta_k, R) \quad \text{und} \quad k = l+1,$$

oder

$$\delta(p, \alpha_{l+1}) = (q, \beta_{k+2}, L) \quad \text{und} \quad k = l-1.$$

Eine *Berechnung* der Turing–Maschine $M$ ist eine Folge von Konfigurationen, wobei jeweils die $(i+1)$–te Konfiguration aus der $i$–ten Konfiguration hervorgeht.

$M$ akzeptiert das Eingabewort $x \in \Sigma^*$, wenn die Berechnung ausgehend von der Konfiguration $q_0 x$ nach endlich vielen Schritten zu einer Konfiguration führt, in welcher der Zustand $q$ ein Element von $F$ ist. ■

**Beispiel 1.4.2:** Auf der Eingabe 0011 etwa führt die Turing–Maschine $M_1$ folgende Berechnung aus:

$$\begin{aligned}
& q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash \\
& q_2 X 0 Y 1 \vdash X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X X Y q_1 1 \vdash \\
& X X q_2 Y Y \vdash X q_2 X Y Y \vdash X X q_0 Y Y \vdash X X Y q_3 Y \vdash \\
& X X Y Y q_3 \vdash X X Y Y \sqcup q_4.
\end{aligned}$$

Die Eingabe 0011 wird also von $M_1$ akzeptiert. ■

**Def. 1.4.3:** Sei $M$ eine Turing–Maschine . $L(M)$, die *von $M$ akzeptierte Sprache*, ist die Menge aller Eingabewörter $x$ (Folgen von Symbolen aus dem Eingabealphabet $\Sigma$), welche von $M$ akzeptiert werden. ■

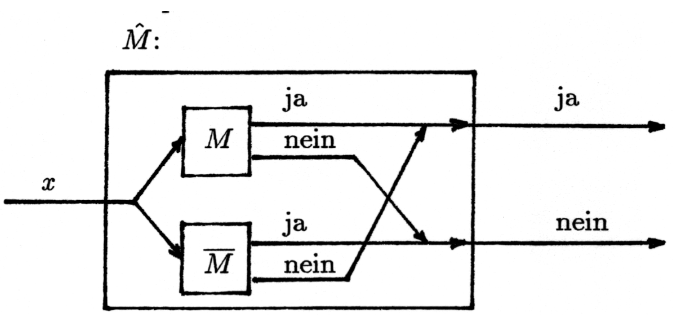
Also $L(M_1) = \{\ 0^n 1^n \ \mid\ n \geq 1\ \}$.

**Def. 1.4.4:** Eine Sprache $L$ (Menge von Wörtern über einem Alphabet $\Sigma$) heißt *rekursiv aufzählbar (r.a.)*, wenn $L = L(M)$ für eine Turing–Maschine $M$. Eine r.a. Sprache $L$, welche von einer Turing–Maschine $M$ akzeptiert wird, die auf jedem Eingabewort stoppt, heißt eine *rekursive* Sprache. In diesem Fall sagt man, $M$ *erkennt* $L$. ■

**Satz 1.4.1:** *Die rekursiven Sprachen sind genau diejenigen Sprachen $L$, für die sowohl $L$ als auch das Komplement $\overline{L}$ von $L$ r.a. sind.*

*Beweis:* Sei $L$ rekursiv. Sei $M$ ein TM, welche immer terminiert und $L$ akzeptiert. Dreht man die Antworten "ja/nein" von $M$ einfach um, so erhält man eine TM $\overline{M}$, welche $\overline{L}$ akzeptiert.

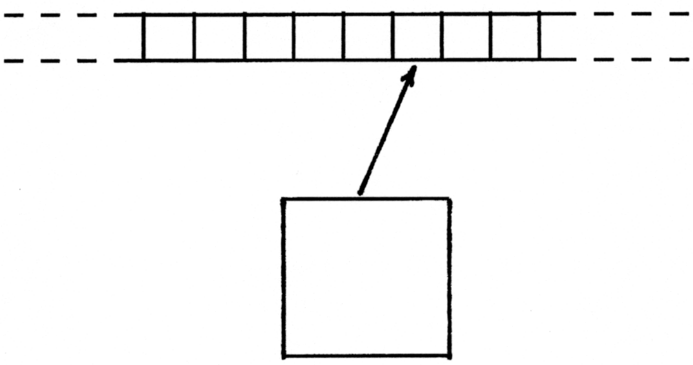
Umgekehrt seien sowohl $L$ als auch $\overline{L}$ r.a. Seien $M, \overline{M}$ TMen, welche $L$ bzw. $\overline{L}$ akzeptieren. Durch Parallelschaltung von $M$ und $\overline{M}$ erhält man eine TM $\hat{M}$, welche immer terminiert und $L$ akzeptiert.




Modifikationen von Turing–Maschinen

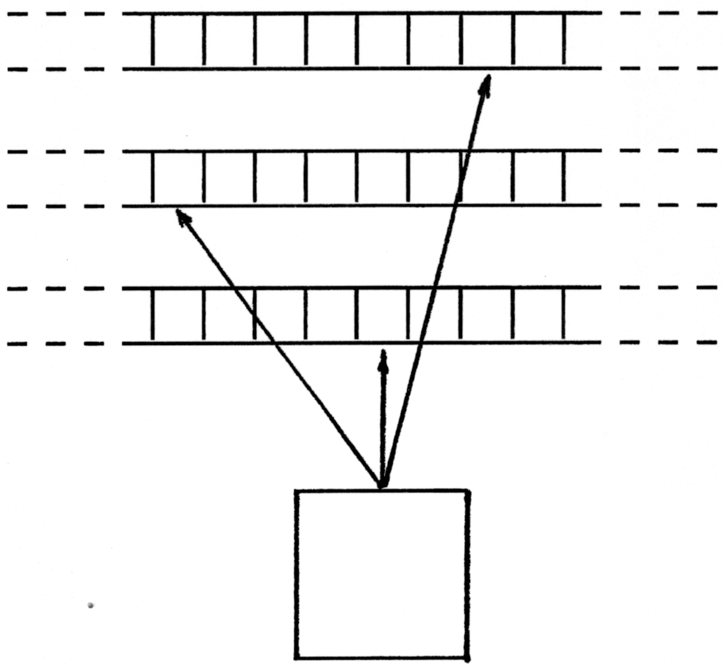
Das Konzept einer Turing–Maschine bietet sich an für eine ganze Reihe von Modifikationen, von denen wir einige kurz beschreiben wollen.

In beide Richtungen unendliches Band:



Das Band ist nicht nur nach rechts unendlich, sondern in beide Richtungen. Ansonsten ändert sich nichts an der Definition der Maschine. Bei Beginn der Berechnung steht das Eingabewort $x$ auf dem Band, das ansonsten nur $\sqcup$'s enthält. Der LS–Kopf zeigt dabei auf das erste Symbol von $x$.

Turing–Maschine mit mehreren Bändern:



Anstatt eines einzigen Bandes hat die Maschine eine endliche Anzahl von Bändern zur Verfügung. Bei Beginn der Berechnung steht das Eingabewort $x$ auf dem ersten Band, das ansonsten nur $\sqcup$'s enthält. Ebenso enthalten alle übrigen Bänder am Beginn der Berechnung nur $\sqcup$'s.

Abhängig vom Zustand der Kontrolleinheit und von den Symbolen, auf welche die LS–Köpfe zeigen, führt die Maschine $M$ in einem Berechnungsschritt folgende Operationen aus:

– $M$ nimmt einen neuen Zustand an,
– schreibt ein Bandsymbol in die Arbeitszelle eines jeden Bandes, und
– bewegt den LS–Kopf eines jeden Bandes unabhängig um eine Position nach rechts oder nach links, oder läßt ihn stationär.

Nichtdeterministische Turing–Maschine:

Abhängig vom Zustand und vom Symbol in der Arbeitszelle hat die Maschine endlich viele Möglichkeiten einen neuen Zustand anzunehmen, ein Symbol in die Arbeitszelle zu schreiben und den LS-Kopf zu bewegen.

**Definition 1.4.5:** Eine *nicht–deterministische Turing–Maschine* (kurz *nTM*) $M$ ist definiert wie ein (deterministische) Turing–Maschine , nur liefert $\delta$ ausgehend von $q$

und $\alpha$ eine endliche Menge möglicher Antworten,

$$\delta(q,\alpha) = \{(p_1,\beta_1,X_1),\ldots,(p_m,\beta_m,X_m)\}.$$

Eine Konfiguration geht aus einer anderen hervor, wenn es eine mögliche Antwort der Überführungsfunktion gibt, mittels derer der Übergang hergestellt werden kann. Ausgehend von einer Konfiguration erhalten wir also nicht eine lineare Berechnung, sondern einen Berechnungsbaum.

$M$ akzeptiert die Eingabe $x$, wenn der Berechnungsbaum ausgehend von der Konfiguration $q_0x$ eine Konfiguration enthält, in welcher der Zustand der Kontrolleinheit $q$ ein Element von $F$ ist. ■

**Beispiel 1.4.3:** Sei $L$ die Sprache

$$L = \{\, 0^n, (01)^n \mid n \geq 1\,\}.$$

Die nichtdeterministische Turing–Maschine $M = (Q,\Sigma,\Gamma,q_0,F,\delta)$ akzeptiert $L$, wobei

$$\begin{aligned} Q &= \{q_0,q_1,q_2,q_3,q_4\},\\ \Sigma &= \{0,1\},\\ \Gamma &= \{0,1,\sqcup\},\\ F &= \{q_4\}, \end{aligned}$$

und $\delta$ durch folgende Tabelle gegeben ist:

| | $\sqcup$ | $0$ | $1$ |
|---|---|---|---|
| $q_0$ | – | $\{(q_1,0,R),(q_2,0,R)\}$ | – |
| $q_1$ | $\{(q_4,\sqcup,R)\}$ | $\{(q_1,0,R)\}$ | – |
| $q_2$ | – | – | $\{(q_3,1,R)\}$ |
| $q_3$ | $\{(q_4,\sqcup,R)\}$ | $\{(q_2,0,R)\}$ | – |
| $q_4$ | – | – | – |

■

**Satz 1.4.2:** *Sei $M$ eine nichtdeterministische Turing–Maschine, welche die Sprache $L$ akzeptiert, also $L = L(M)$. Dann gibt es eine deterministische Turing–Maschine $M'$, welche ebenfalls $L$ akzeptiert, also $L = L(M')$.*

*Beweis:* Für jedes Paar bestehend aus einem Zustand und einem Bandsymbol von $M$ gibt es eine endliche Anzahl von Wahlmöglichkeiten in der Überführungsfunktion von $M$. Wir numerieren diese Möglichkeiten durch 1,2,... . Sei $r$ die größte Anzahl von

Wahlmöglichkeiten für ein solches Paar. Dann kann jede endliche Folge von Wahlen dargestellt werden als eine Folge der Nummern 1 bis $r$. Natürlich entspricht nicht jede Folge solcher Nummern einer Folge von Wahlen, da es für ein gegebenes Paar eventuell weniger als $r$ Wahlmöglichkeiten geben kann.

Die deterministische Turing–Maschine $M'$ hat nun drei Bänder. Das erste Band ist das Eingabeband. Auf dem zweiten Band erzeugt $M'$ systematisch endliche Folgen über $\{1, \ldots, r\}$. Dabei werden zunächst die Folgen der Länge 1 erzeugt, dann diejenigen der Länge 2, usw.

Für jede Folge, welche auf Band 2 erzeugt wird, kopiert $M'$ die Eingabe auf Band 3 und simuliert $M$ auf Band 3, wobei die Folge auf Band 2 dazu benutzt wird, um die Wahlen der entsprechenden Zweige des Berechnungsbaums durchzuführen. Gibt es im Berechnungsbaum von $M$ einen Zweig, der zu einem akzeptierenden Zustand führt, so wird $M'$ schließlich eine Kodierung dieses Zweiges auf Band 2 erzeugen, und bei der Simulierung dieses Zweiges wird $M'$ die Eingabe akzeptieren. Gibt es im Berechnungsbaum von $M$ keinen Zweig, der zu einem akzeptierenden Zustand führt, so wird auch $M'$ nicht akzeptieren.

Also $L(M) = L = L(M')$. ■

Wir haben also nun einige Möglichkeiten zur Modifikation von Turing–Maschinen betrachtet. Keine dieser Möglichkeiten ist jedoch in der Lage, die Berechnungspotenz einer Turing–Maschine zu steigern. Eine Sprache $L$, welche von einer auf diese Weise modifizierten Turing–Maschine akzeptiert wird, kann auch von einer Turing–Maschine im ursprünglichen Sinn akzeptiert werden. Für eine genauere Betrachtung dieser Berechnungsäquivalenz verweisen wir auf [HU].

Turing–Maschinen als Generatoren von Sprachen

**Definition 1.4.6:** Sei $M$ eine Turing–Maschine mit einem oder mehreren Bändern, welche das erste Band als Ausgabeband benützt. $M$ hat ein ausgezeichnetes Bandsymbol #, und $M$ bewegt den LS–Kopf des Ausgabebandes niemals nach links. In diesem Fall ist $M$ geeignet eine Sprache zu erzeugen (generieren). Und zwar ist die *von $M$ erzeugte (generierte) Sprache $G(M)$* die Menge aller Wörter, welche $M$ ohne irgend eine Eingabe zwischen zwei #'s auf das Ausgabeband schreibt. ■

**Satz 1.4.3:** *Eine Sprache $L$ über einem Alphabet $\Sigma$ ist genau dann r.a., wenn $L$ von einer Turing–Maschine generiert werden kann. (Daher auch die Bezeichnung "rekursiv* *aufzählbar".)*

*Beweis:* Wir wollen einen Beweis dieser Behauptung skizzieren. Angenommen $L$ wird von der Turing–Maschine $M$ generiert, also $G(M) = L$. Wir konstruieren eine Turing–Maschine $M'$ mit einem Band mehr als $M$. $M'$ simuliert $M$. Immer wenn $M$ ein Wort $x$ erzeugt, so vergleicht es $M'$ mit dem Eingabewort. Stimmen sie überein, so akzeptiert $M'$ das Eingabewort. Also $L(M') = L$.

Andererseits sei $M$ eine Turing–Maschine, welche $L$ akzeptiert, also $L(M) = L$. Wir konstruieren eine Turing–Maschine $M'$, welche nacheinander alle Paare von positiven ganzen Zahlen $(i, j)$ (in kodierter Form) erzeugt. Sobald das Paar $(i, j)$ erzeugt worden ist, erzeugt $M'$ das $i$–te Wort $x_i$ über dem Alphabet $\Sigma$ (in dieser Aufzählung sind die Wörter der Länge nach und Wörter gleicher Länge alphabetisch geordnet) und simuliert $j$ Berechnungsschritte von $M$ auf der Eingabe $x_i$. Wird $x_i$ von $M$ in nicht mehr als $j$ Schritten akzeptiert, so erzeugt $M'$ das Wort $x_i$. Somit ist $L = G(M')$. ■

Turing–Maschinen als Berechner von Funktionen

Turing–Maschinen können nicht nur zum Erkennen und Erzeugen von Sprachen eingesetzt werden, sondern sie sind auch in der Lage Funktionen zu berechnen.

**Definition 1.4.7:** Seien $\Sigma_1$ und $\Sigma_2$ zwei Alphabete und $f$ eine (partielle) Funktion von $(\Sigma_1^*)^k$ in $\Sigma_2^*$ ($\Sigma_1^*$ ist die Menge aller Wörter über $\Sigma_1$). Eine Turing–Maschine $M$ *berechnet* die Funktion $f$ falls

- $\Sigma_1 \cup \Sigma_2$ eine Teilmenge des Bandalphabets von $M$ ist, und
- für jedes $x = (x_1, \ldots, x_k) \in (\Sigma_1^*)^k$, wenn $f(x_1, \ldots, x_k) = y$ dann stoppt $M$ auf dem Eingabewort $x_1 \sqcup \ldots \sqcup x_k$ mit Ausgabe (Bandinhalt nach Ende der Berechnung) $y$ und wenn $f(x_1, \ldots, x_k)$ undefiniert ist, dann stoppt $M$ auf dem Eingabewort $x_1 \sqcup \ldots \sqcup x_k$ nicht.

Eine (partielle) Funktion, welche von einer Turing–Maschine berechnet werden kann, heißt *(partiell) Turing–berechenbar*, abgekürzt *(p.)T.b.* ■

**Beispiel 1.4.4:** Ein Beispiel einer T.b. Funktion ist die modifizierte Differenz $\dot{-}$, welche für zwei natürliche Zahlen $m$, $n$ den Wert $m - n$ liefert falls $m \geq n$ und den Wert 0 sonst. $\dot{-}$ wird von der Turing–Maschine $M_2$ berechnet mit

Zustandsmenge $\ldots . Q = \{q_0, \ldots, q_6\}$,
Eingabealphabet $\ldots \Sigma = \{0\}$,

Bandalphabet ..... $\Gamma = \{\sqcup, 0, 1\}$,
Endzustände ...... $F = \{\}$.

Die Eingabeparameter $m$, $n$ werden kodiert als

$$\underbrace{0\ldots0}_{m \text{ mal}} \sqcup \underbrace{0\ldots0}_{n \text{ mal}}.$$

Die Überführungsfunktion $\delta$ von $M_2$ arbeitet wie folgt:

(1) Start. Ersetze die führende 0 durch $\sqcup$. Ist keine 0 vorhanden, so lösche den Rest des Bandes.
$\delta(q_0, 0) = (q_1, \sqcup, R)$, $\delta(q_0, \sqcup) = (q_5, \sqcup, R)$.

(2) Bewege den LS–Kopf nach rechts und suche nach dem ersten $\sqcup$. Überschreibe dieses durch 1.
$\delta(q_1, 0) = (q_1, 0, R)$, $\delta(q_1, \sqcup) = (q_2, 1, R)$, $\delta(q_1, 1) = (q_2, 1, R)$.

(3) Bewege den LS–Kopf nach rechts über 1en hinweg bis zur ersten 0. Überschreibe diese durch 1.
$\delta(q_2, 1) = (q_2, 1, R)$, $\delta(q_2, 0) = (q_3, 1, L)$.

(4) Gehe nach links bis zum ersten $\sqcup$. Nimm Zustand $q_0$ an und wiederhole den Zyklus.
$\delta(q_3, 0) = (q_3, 0, L)$, $\delta(q_3, 1) = (q_3, 1, L)$, $\delta(q_3, \sqcup) = (q_0, \sqcup, R)$.

(5) Wird im Zustand $q_2$ ein $\sqcup$ vor einer 0 gefunden, so nimm Zustand $q_4$ an und gehe nach links, wobei alle 1en durch $\sqcup$'s überschrieben werden, bis zum ersten $\sqcup$. Ändere dieses $\sqcup$ wieder zu einer 0 ab, nimm Zustand $q_6$ an und stoppe.
$\delta(q_2, \sqcup) = (q_4, \sqcup, L)$, $\delta(q_4, 1) = (q_4, \sqcup, L)$, $\delta(q_4, 0) = (q_4, 0, L)$, $\delta(q_4, \sqcup) = (q_6, 0, R)$.

(6) Wird im Zustand $q_0$ eine 1 anstatt einer 0 gefunden, so ist offensichtlich der erste Block von 0en abgearbeitet. Nimm Zustand $q_5$ an und lösche den Rest des Bandes. Anschließend nimm Zustand $q_6$ an und stoppe.
$\delta(q_0, 1) = (q_5, \sqcup, R)$, $\delta(q_5, 0) = (q_5, \sqcup, R)$, $\delta(q_5, 1) = (q_5, \sqcup, R)$, $\delta(q_5, \sqcup) = (q_6, \sqcup, R)$.

| Zustand | Symbol | | |
|---|---|---|---|
| | 0 | 1 | $\sqcup$ |
| $q_0$ | $(q_1, \sqcup, R)$ | $(q_5, \sqcup, R)$ | $(q_5, \sqcup, R)$ |
| $q_1$ | $(q_1, 0, R)$ | $(q_2, 1, R)$ | $(q_2, 1, R)$ |
| $q_2$ | $(q_3, 1, L)$ | $(q_2, 1, R)$ | $(q_4, \sqcup, L)$ |

| Zustand | Symbol 0 | Symbol 1 | Symbol $\sqcup$ |
|---|---|---|---|
| $q_3$ | $(q_3, 0, L)$ | $(q_3, 1, L)$ | $(q_1, \sqcup, R)$ |
| $q_4$ | $(q_4, 0, L)$ | $(q_4, \sqcup, L)$ | $(q_6, 0, R)$ |
| $q_5$ | $(q_5, \sqcup, R)$ | $(q_5, \sqcup, R)$ | $(q_6, 0, R)$ |
| $q_6$ | $-$ | $-$ | $-$ |

Berechnung von $M_2$ auf $00 \sqcup 0 \quad (2 \dot{-} 1)$:

$$q_0 00 \sqcup 0 \vdash \sqcup q_1 0 \sqcup 0 \vdash \sqcup 0 q_1 \sqcup 0 \vdash \sqcup 01 q_2 0 \vdash$$
$$\sqcup\, 0 q_3 11 \vdash \sqcup q_3 011 \vdash q_3 \sqcup 011 \vdash \sqcup q_0 011 \vdash$$
$$\sqcup \sqcup q_1 11 \vdash \sqcup \sqcup 1 q_2 1 \vdash \sqcup \sqcup 11 q_2 \vdash \sqcup \sqcup 1 q_4 1 \vdash$$
$$\sqcup \sqcup q_4 1 \vdash \sqcup q_4 \vdash \sqcup 0 q_6.$$

Berechnung von $M_2$ auf $0 \sqcup 00 \quad (1 \dot{-} 2)$:

$$q_0 0 \sqcup 00 \vdash \sqcup q_1 \sqcup 00 \vdash \sqcup 1 q_2 00 \vdash \sqcup q_3 110 \vdash$$
$$q_3 \sqcup 110 \vdash \sqcup q_0 110 \vdash \sqcup \sqcup q_5 10 \vdash \sqcup \sqcup \sqcup q_5 0 \vdash$$
$$\sqcup \sqcup \sqcup \sqcup q_5 \vdash \sqcup \sqcup \sqcup \sqcup \sqcup q_6.$$

■

Alle üblichen arithmetischen Funktionen wie $m + n$, $m \cdot n$, $m^n$, $n!$, $\lceil \log_2 n \rceil$, etc. sowie Kombinationen davon sind T.b.

Zwischen r.a. Sprachen und p.T.b. Funktionen besteht ein enger Zusammenhang.

**Satz 1.4.4:** *Ist $f$ eine partielle Funktion von $(\Sigma_1^*)^k$ nach $\Sigma_2^*$, so ist $f$ genau dann p.T.b., wenn $L_f = \{ (x_1, \ldots, x_k, y) \mid f(x_1, \ldots, x_k) = y \}$ r.a. ist.*

*Beweis:* Übung. ■

Im folgenden befassen wir uns mit der Äquivalenz zwischen dem RAM/RASP Modell und dem Turing–Maschinen Modell.

**Satz 1.4.5:** *Jede Turing–Maschine (mit mehreren Bändern) kann durch eine RAM simuliert werden. Also jede Sprache (Funktion), die von einer Turing–Maschine akzeptiert (berechnet) werden kann, kann auch von einer RAM akzeptiert (berechnet) werden.*

*Beweis:* (Skizze) Sei $M$ eine $k$–Band Turing–Maschine . In der $M$ simulierenden RAM $R$ ist jede Bandzelle von $M$ in einem Register abgespeichert, und zwar ist die $i$–te Zelle des $j$–ten Bandes im Register $ki + j + c$ abgespeichert. Dabei ist $c$ so angelegt, daß $R$ noch Arbeitsspeicher zur Verfügung hat, u.a. zur Abspeicherung der Positionen der $k$ LS–Köpfe von $M$. Bandzellen von $M$ können von $R$ gelesen werden durch indirekte Adressierung mittels der Register, welche die Positionen der LS–Köpfe beinhalten. ■

**Satz 1.4.6:** *Jede RAM kann durch eine Turing–Maschine simuliert werden. Also jede Sprache (Funktion), die von einer RAM akzeptiert (berechnet) werden kann, kann auch von einer Turing–Maschine akzeptiert (berechnet) werden.*

**Beweis:** (Skizze) Die Turing–Maschine $M$, die die RAM $R$ simuliert, besitzt neben den Ein– und Ausgabebändern noch drei zusätzliche Bänder $B_3, B_4, B_5$. $B_3$ dient zur Darstellung des Speichers von $R$ und hat die Form

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | # | # | $i_1$ | # | $c_1$ | # | # | $i_2$# | $c_2$ | # | # | $\cdots$ | $i_k$ | # | $c_k$ | # | # | $\sqcup$ | ... |

Dabei sind $i_1, \ldots, i_k$ die Indizes derjenigen Register von $R$, welche nicht 0 enthalten und $c_j = c(i_j)$; $i_j, c_j$ sind in Binärdarstellung auf $B_3$ wiedergegeben, für $1 \leq i \leq k$. Der Inhalt des Akkumulators von $R$ ist in Binärdarstellung auf $B_4$ dargestellt, und $B_5$ wird als Arbeitsband verwendet.

Jeder Rechenschritt von $R$ entspricht einer endlichen Folge von Zuständen der Turing–Maschine $M$. Als Beispiel betrachten wir die RAM Instruktionen `ADD *i` und `STORE i`. Die Instruktion `ADD *i` wird von $M$ folgendermaßen simuliert:

1. Suche auf $B_3$ nach einer Eintragung für das RAM Register $i$, d.h. eine Folge $\#\#i'\#$, wobei $i'$ die Binärdarstellung von $i$ ist. Ist eine solche Eintragung vorhanden, so schreibe die nächste Zahl, welche der Inhalt von Register $i$ ist, auf $B_5$; andernfalls halte (der Inhalt von Register $i$ ist 0, die indirekte Adressierung kann also nicht durchgeführt werden).
2. Suche auf $B_3$ nach einer Eintragung für das RAM Register, dessen Index auf $B_5$ steht. Ist eine solche Eintragung vorhanden, so schreibe den Inhalt dieses Registers auf $B_5$; andernfalls schreibe 0 auf $B_5$.
3. Addiere die Zahl auf $B_5$ zum Inhalt des Akkumulators, der auf $B_4$ steht.

Die Instruktion `STORE i` wird von $M$ folgendermaßen simuliert:

1. Suche auf $B_3$ nach einer Eintragung für das RAM Register $i$, also $\#\#i'\#$.

2. Ist eine solche Eintragung vorhanden, so kopiere den Bandinhalt von $B_3$ rechts von $\#\#i'\#$, außer der unmittelbar nächsten Zahl ($c(i)$), auf $B_5$. Kopiere den Inhalt von $B_4$ (Akkumulator) auf $B_3$ unmittelbar rechts von $\#\#i'\#$ und unmittelbar daran den Inhalt von $B_5$.
3. Ist eine Eintragung für Register $i$ nicht vorhanden, so schreibe $i'\#$ auf $B_3$ unmittelbar nach dem letzten von $\sqcup$ verschiedenen Symbol, anschließend den Inhalt von $B_4$ (Akkumulator) und schließlich $\#\#$.

Auf ganz ähnliche Weise können auch die übrigen RAM Instruktionen simuliert werden.

■

Die Menge der (partiell) T.b. (zahlentheoretischen) Funktionen ist nicht nur identisch mit der Menge der (partiell) RAM berechenbaren Funktionen, sie ist auch identisch mit der Menge der (partiell) rekursiven Funktionen, oder mit der Menge der WHILE-berechenbaren Funktionen (siehe etwa [DW], [Sch]). Alle diese mathematischen Modelle liefern dieselbe Klasse algorithmischer Funktionen. Diese Klasse scheint also in gewisser Weise eine natürliche zu sein. Andererseits sind alle (bisher betrachteten) Funktionen, die man intuitiv als berechenbar bezeichnen würde, tatsächlich rekursiv. Die meisten Mathematiker und Computerwissenschafter sind daher der Ansicht, daß mit dem formalen Begriff der rekursiven bzw. Turing–berechenbaren Funktion der intuitive Begriff der "algorithmischen" Funktion charakterisiert ist. Diese Überzeugung wurde zum ersten Mal von A. Church ausgesprochen und heißt nach ihm die *Church'sche These*:

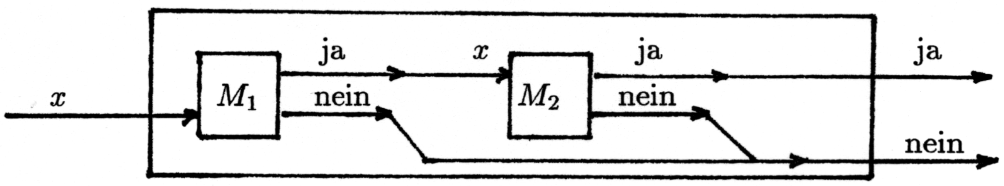
> *Eine Funktion ist berechenbar genau dann wenn sie Turing–berechenbar (rekursiv) ist.*

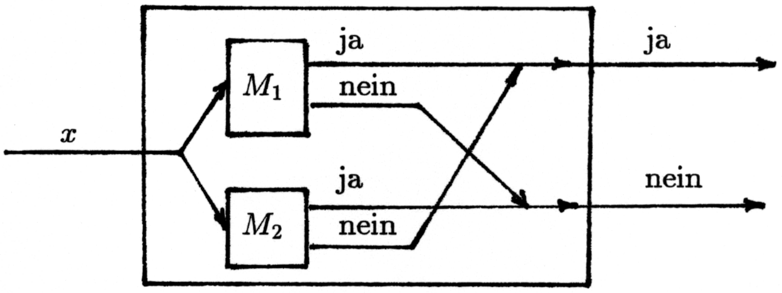
Im strengen Sinne beweisen läßt sich die Church'sche These natürlich nicht, da man auf einem intuitiven Begriff keinen Beweis aufbauen kann. Aufgrund der Church'schen These versteht man heutzutage unter einem Algorithmus eine Turing–Maschine . Wenn wir also sagen, daß sich ein Problem algorithmisch lösen läßt, so meinen wir streng genommen damit, daß es eine Turing–Maschine gibt, die das Problem löst.

**Definition 1.4.8:** Ein *Algorithmus* ist eine Turing–Maschine , die auf jeder Eingabe terminiert. ■

Bisher haben wir nur Algorithmen definiert, welche ganze Zahlen als Eingaben nehmen und wiederum ganze Zahlen produzieren. Im intuitiven Sinne kennt man natürlich auch Algorithmen mit ganz anderen Ein– und Ausgabebereichen. Zur Formalisierung solcher Algorithmen gibt es im Prinzip zwei Ansätze. Beim ersten Ansatz ordnet man jedem Element der Ein– und Ausgabemenge eine natürliche Zahl zu und kann dann das herkömmliche Konzept des zahlentheoretischen Algorithmus verwenden. Beim zweiten Ansatz verlangt man nur, das sich die Elemente der Ein– und Ausgabemenge als Wörter über einem Alphabet schreiben lassen. Solche Wörter können nun von einer Turing–Maschine gelesen bzw. geschrieben werden und man erhält auf diese Weise eine Verallgemeinerung des Begriffs "Algorithmus".

**Übung:** Seien $M_1, M_2$ Turing–Maschinen. Dann gibt es Turing–Maschinen $M_s, M_p$ mit dem Verhalten




**Übung:** Das Komplement einer rekursiven Sprache ist rekursiv.

**Übung:** Die Vereinigung zweier rekursiver Sprachen ist rekursiv. Die Vereinigung zweier r.a. Sprachen ist r.a.