# Theoretische Informatik II

## Script zur Vorlesung vom 01.12.20000

Angefertigt von: Tony Schmidt
Matrikel-Nr.: 702781

## Inhaltsverzeichnis

**Inhaltsverzeichnis ..... 1**

**6.0. Einleitung ..... 2**

**6.1. Turing-Maschine ..... 2**

6.1.1 Warum Turing-Maschinen? ..... 2
6.1.2 Beschreibung der Turing-Maschine ..... 2
6.1.3 Arbeitsweise der Turing-Maschine ..... 3
6.1.4 Definition A: *Turing-Maschine t* ..... 3
6.1.5 Definition B: *Konfiguration* ..... 3
6.1.6 Definition C: *Arbeitsweise (Konfiguration)* ..... 4
6.1.7 Definition D: *t berechnete Funktion* ..... 5
6.1.8 Definition E: *Turing-berechenbar (+Klassen)* ..... 5
6.1.9 Beispiele ..... 6
6.1.10 Varianten von Turing-Maschinen ..... 7
6.1.11 Universelle Turing-Maschine* ..... 8

**6.2. Registermaschine ..... 9**

6.2.1 Warum Registermaschinen? ..... 9
6.2.2 Beschreibung der Registermaschine ..... 9
6.2.3 Definition F: *k-Registermaschine r* ..... 9
6.2.4 Definition G: *Konfiguration* ..... 10
6.2.5 Definition H: *$RM_k$-berechenbar (+Klassen)* ..... 11
6.2.6 Beispiele ..... 11
6.2.7 RAM-random access machine* ..... 12

**Anhang:** Zusammenfassung ..... 13
Ausblick auf die nächste Vorlesung (nächste Script) ..... 13
Literaturverzeichnis ..... 13

* nicht aus der Vorlesung

**Einleitung**

Dieses Script ist durch eine Mitschrift der Vorlesung Theoretische Informatik II am 01.12.2000 entstanden. Diese würde jedoch mit der Definition E zu Turing- Maschinen beginnen. Um das Verständnis des Stoffes zu verbessern, habe ich den Stoff über Turing-Maschinen aus der vorherigen Vorlesung am 24.11.2000 hier mit aufgenommen. Vorraussetzung ist natürlich auch das Wissen des bis hierhin bekannten Stoffes.
Dies ist eine überarbeitete Fassung und sollte fehlerfrei sein, garantiert kann dies jedoch nicht, da auch Tippfehler übersehen werden können. Also immer Mitdenken!
In der Vergangenheit haben wir uns mit kontextfreien Grammatiken beschäftigt. Da diese mit Einschränkungen verbunden sind, betrachten wir nun *Grammatiken vom Typ 0* und die damit verbundenen *aufzählbaren Sprachen Ch-0*. Jetzt sind aber Maschinenmodelle interessant, da sie den verwendeten Rechner (Maschinen) ihre Leistungsfähigkeit entsprechen. Man kann ohne Einschränkung der Leistungsfähigkeit einen realen Rechner auf ein Maschinenmodell abstrahieren. Ein Rechner benötig immer Speicher zum Rechnen, so dass man grundlegend zwei Maschinenmodelle (*Turing-Maschine* und *Registermaschine*) erhält, die im folgenden hier definiert werden, sowie ihre Arbeitweisen beschrieben werden.

# 6.0 Turingmaschinen

1936 von Adam Mathison Turing (1912-1954) entwickelt.

## 6.1.1 Warum Turing-Maschinen?

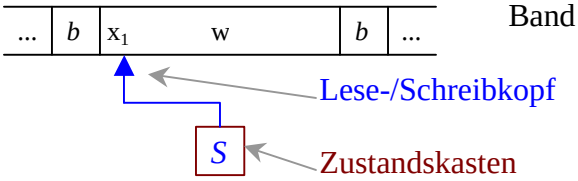
Endliche Automaten besitzen nur einen einzigen Speicher, der endlich viele Information speichern kann. In der Praxis sind alle Speicher endlich, so dass durch diese Beschränkung der endliche Automat sehr begrenzte Berechnungsmöglichkeiten hat. *Push-Down-Akzeptoren* besitzen auch einen endlich großen Zustandsspeicher, verfügen jedoch über einen potentiellen unendlich großen *Stack*. Ein *Stack* erlaubt nur *LIFO*-Zugriff (last in first out), womit er nur das oberste Element sehen kann. Nun definiert man eine Maschine, die einen endlichen Zustandsraum sowie einen potentiell endlichen Speicher ohne *LIFO*-Beschränkung besitzt. Frage ist nun, ob er dadurch an Mächtigkeit gewinnt. Man wird feststellen das dies so ist. Diese Maschinen nennt man Turing-Maschinen.

## 6.1.2 Beschreibung der Turing-Maschine

*Turing-Maschinen* können also *endlich viele Zustände* einnehmen und besitzen zusätzlich einen *potentiell unendlich großen Speicher*. Diesen kann man sich als ein unendlich langes Band vorstellen. Der Zugriff erfolgt über ein Schreib-/Lesekopf, der nach rechts (*r*) oder nach links (*l*) verschoben werden kann. Die Maschine kann ein Eingabewort auf dem Band lesen und sie kann auch Zeichen aus einen endlichen Alphabet auch schreiben. Da man im Gegensatz zu *Push-Down-Akzeptoren* durch die Linksbewegung ein Eingabewort mehrmals lesen kann, muss man noch ein Haltezustand (*h*) einführen. Dieser kann im Laufe einer Berechnung (höchstens) ein einziges mal auftreten, am Berechnungsende nämlich. Man führt auch ein sogenanntes *Blanksymbol* (*b*) aus den endlichen Bandalphabet ein, um u. a. Wörter abzugrenzen.

### 6.1.3 Arbeitsweise einer Turing-Maschine

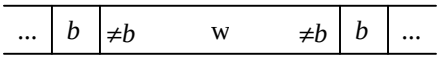
a) *Initialisierung:* $w \in X^*, \quad w = x_1 \ldots x_n, \quad x_i \in X \subset G$ $X \mathrel{\hat{=}}$ Eingabealphabet (endl.)
$G \mathrel{\hat{=}}$ Bandalphabet (endl.)
$S \mathrel{\hat{=}}$ Zustandsmenge (endl.)




b) *Schritt ausführen:* (Zeichen unter Lese-/Schreibkopf (LSK), aktueller Zustand) $\overset{d}{\rightarrow}$
(neuer Zustand, neues Zeichen unter LSK, LSK-Bewegung ($l$,$r$,$h$) )

z.B. $d(s,a) = (s',a',P); \quad s,s' \in S; \quad a,a' \in G; \quad P \in \{l,r,h\}$

c) *Berechnungsende:*




$\Rightarrow$ Ausgabe: $w$

### 6.1.4 Definition A:

Ein 6-Tupel $t = (S, X, G, d, s_0, b)$ heißt *Turing-Maschine*, wenn gilt

- $S$ nichtleere endliche Zustandsmenge
- $s_0 \in S$ Anfangszustand
- $G$ endliches Bandalphabet
- $X \subset G$ endliches Eingabealphabet
- $b \in G \setminus X$ Blanksymbol
- $d : S \times G \rightarrow S \times G \times \{l,r,h\}$ partielle Zustandsüberführungsfunktion

### 6.1.5 Definition B:

Ein Tripel $(s, f, i)$ heißt Konfiguration der Turing-Maschine, wenn

- $s \in S$ aktueller Zustand
- $f: \ddot{u} \rightarrow G$ Bandinhalt: *f(j)* ist der Inhalt der *j*-ten Zelle, wobei es nur endlich viele *j* mit *f(j)*¹*b* gibt.
- $i \in \ddot{u}$ Position des Lese-/Schreibkopf

erfüllt ist.

$K_t$ ist dabei die Menge aller Konfigurationen von $t$ .

### 6.1.6 Definition C:

Arbeitsweise einer Turing-Maschine anhand einer abstrakten Automatensymbolik.

(1) *Anfangskonfiguration*: $w \in X^*, \quad w = w_0 ... w_k, \quad w_i \in X$

$a : X^* \to K_t$

$a(w) = (s_0, f_w, d)$ $f_w$ ist die Bandinschrift

$$f_w(j) = \begin{cases} w_j & , falls \ 0 \le j \le k \\ b & , sonst \end{cases}$$

*Start:* Auf dem Band steht ein Eingabewort $w \in X^*$, alle Felder links und rechts von *w* enthalten das Blanksymbol *b*. Der Lese-/Schreibkopf steht auf dem ersten Buchstabe von *w* und die Turing-Maschine befindet sich im Anfangszustand $s_0$.

(2) *Zustandsübergangskonfiguration* (Schritt ausführen):

$\hat{d} : K_t \to K_t$ mit

$\hat{d}(s, f, i) = (s', f', i') :\Leftrightarrow f(i) = a \in G, \ d(s, a) = (s', a', P)$

$$f'(j) = \begin{cases} f(j) & , falls \ j + i \\ a' & , falls \ j = i \end{cases} \qquad i' = \begin{cases} i+1 & , falls \ P = \{r\} \\ i-1 & , falls \ P = \{l\} \\ i & , falls \ p = \{h\} \end{cases}$$

*Arbeitsschritt:* Der LSK befindet sich auf einem Feld mit dem Inhalt $a \in G$, der Zustandskasten sei im Zustand *s* (nicht Endzustand).
Ist nun $d(s, a) = (s', a', P)$ definiert, geht die Maschine in de Zustand *s'* über.
z.B. wird das Zeichen *a* auf dem aktuellen Feld des Bandes durch *a'* ersetzt und der LSK kann dann nach rechts *r* verschoben werden.

(3) *Ausgabefunktion* (Ergebnis):

$w : K_t \to G^*$ mit

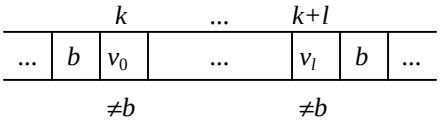
$w(s, f, i) = e$ , falls für alle $j \in ü$ $f(j)=b$

$w(s, f, i) = v_0 v_1 ... v_l$ , falls es ein $k \in ü$ gibt mit

$f(j) = b$ für alle $j < k \quad f(k) \ne b$

$f(j) = b$ für alle $j > k + l \quad f(k + l) \ne b$

$f(j) = v_{j-k}$ für $k \le j \le k + l$




*Ergebnis:* Die Turing-Maschine hält an, wenn die Berechnung beendet ist.
Die Ausgabe ist dann das Wort zwischen den am weitesten links stehenden Zeichen (ungleich dem Blanksymbol *b*) und dem am weitesten rechts stehenden Zeichen (ungleich *b*).
→ Es gibt genau ein *k*. Somit ist die Ausgabe eindeutig.

### 6.1.7 Definition D:

Sei $t = (S, X, G, \delta, s_0, b)$ eine *Turing-Maschine.*

Die von $t$ *berechnete Funktion* $h_t : X^* \to G^*$ ist definiert durch:

$$h_t(w) = \begin{cases} w(\hat{\delta}^{m+1}(a(w))) & , falls\ m = Min\left\{ j \middle| \hat{\delta}^{j}(a(w)) = (s, f, i)\ und\ \exists s', a' : \delta(s, f(i)) = (s', a', h) \right\} ex. \\ \perp & , sonst\ (Endlosschleife) \end{cases}$$

Diese Funktion ist nur dann definiert, wenn die Turing-Maschine hält, also es eine Berechnungsende gibt.

### 6.1.8 Definition E:

Seien *A,B* Alphabete.

a) Eine Abbildung $h : A^* \to B^*$ (partielle Funktion) heißt *Turing-berechenbar*, wenn es eine Turing-Maschine $t = (S, X, G, \delta, s_0, b)$ gibt mit $B \subseteq G$ und $h_t = h$.

b) Klasse $\tau_{A,B} = \{h \mid h : A^* \to B^*\ \ Turing\text{-}berechenbar\}$

c) $\tau$ sei die Klasse aller *Turing-berechenbaren* Funktionen (*A,B* beliebig)

d) Eine Menge $M \subseteq A^*$ heißt *Haltebereich* / Definitionsbereich einer Turing-Maschine $t = (S, A, G, \delta, s_0, b)$, wenn $M = \{w \in A^* \mid h_t(w) \text{ definiert ist}\}$.

e) Eine Menge $M \subseteq B^*$ heißt *Ergebnisbereich* / Wertebereich einer Turing-Maschine $t = (S, A, G, \delta, s_0, b)$ mit $B \subseteq G$, wenn $M = \{v \in B^* \mid \exists w \in A^* : h_t(w) = v\}$.

### 6.1.9 Beispiele

Im folgenden kommen nun ein paar Beispiele zu Turing-Maschinen in der sogenannten „*Bierdeckelnotation*“ oder auch „*Busy-Beaver-Notation*“ genannt.

**Beispiel 1:** $t_1 = (S_1 = \{s_0\}, X_1 = \{|\}, G_1 = \{b,|\}, d_1, s_0, b)$ mit $P \in \{l,r,h\}$ und $d(s_0,b) = (s_0,|,h)$ und $d(s_0,|) = (s_0,|,r)$.

Es werden Striche auf das Band geschrieben gefolgt von einen Blanksymbol.

| $S_1$ | $G_1$ | $S_1$ | $G_1$ | $P$ |
|---|---|---|---|---|
| $s_0$ | $\|$ | $s_0$ | $\|$ | $r$ |
| $s_0$ | $B$ | $s_0$ | $\|$ | $h$ |

**Tabelle 1**

$h_{t_1} : \{|\}^* \to \{b,|\}^*$

$h_{t_1}(|^n) = |^{n+1}$ , $n \in \hat{u}_0$

Haltebereich: $\{|\}^*$

Wertebereich: $\{|\}^+$

**Beispiel 2:** $t_2 = (S_2 = \{s_0\}, X_2 = \{|\}, G_2 = \{b,|\}, d_2, s_0, b)$ mit $P \in \{l,r,h\}$ und $d(s_0,b) = (s_0,b,h)$ und $d(s_0,|) = (s_0,b,r)$.

Es werden Striche auf dem Band durch Blanksymbole ersetzt.

| $S_2$ | $G_2$ | $S_2$ | $G_2$ | $P$ |
|---|---|---|---|---|
| $s_0$ | $b$ | $s_0$ | $b$ | $r$ |
| $s_0$ | $\|$ | $s_0$ | $b$ | $h$ |

**Tabelle 2**

$h_{t_2} : \{|\}^* \to \{b,|\}^*$

$h_{t_2}(w) = e$ , $w \in \{|\}^*$

Haltebereich: $\{|\}^*$

Wertebereich: $\{e\}$

**Beispiel 3:** $t_3 = (S_3 = \{s_0, s_1\}, X_3 = \{|\}, G_3 = \{b,|\}, d_3, s_0, b)$ mit $P \in \{l,r,h\}$ und der Zustandsübergangsfunktion $d : S \times G \to S \times G \times \{l,r,h\}$ (siehe Tabelle 3).

Es handelt sich hierbei um eine Gradzahligkeitsmaschine, die nur hält, wenn eine gerade Anzahl von Strichen auf dem Band ist und nach dem Lesen ist ein Strich mehr auf dem Band.

| $S_3$ | $G_3$ | $S_3$ | $G_3$ | $P$ |
|---|---|---|---|---|
| $s_0$ | $b$ | $s_1$ | $\|$ | $h$ |
| $s_0$ | $\|$ | $s_1$ | $\|$ | $r$ |
| $s_1$ | $b$ | $s_1$ | $b$ | $r$ |
| $s_1$ | $\|$ | $s_0$ | $\|$ | $r$ |

**Tabelle 3**

$h_{t_3} : \{|\}^* \to \{b,|\}^*$

$$h_{t_3}(|^n) = \begin{cases} |^{n+1} & , falls \ \frac{n}{2} \in \hat{u}_0 \\ \perp & , sonst \end{cases}$$

Haltebereich: $\{|^{2k} \mid k \in \hat{u}_0\}$

Wertebereich: $\{|^{2k+1} \mid k \in \hat{u}_0\}$

**Beispiel 4:** $t_4 = (S_4 = \{s_0, s_1, s_2, s_3\}, X_3 = \{|\}, G_4 = \{b, |, c\}, d_4, s_0, b)$ mit $P \in \{l, r, h\}$ und der Zustandsübergangsfunktion $d : S \times G \rightarrow S \times G \times \{l, r, h\}$ (siehe Tabelle 4).

Die auf dem Band befindlichen Striche werden durch doppelt so viele *c*'s ersetzt (Verdopplung).

| Nr. | $S_4$ | $G_4$ | $S_4$ | $G_4$ | $P$ |
|---|---|---|---|---|---|
| 1) | $s_0$ | $b$ | $s_0$ | $b$ | $h$ |
| 2) | $s_0$ | $\|$ | $s_1$ | $b$ | $r$ |
| 3) | $s_0$ | $c$ | $s_0$ | $c$ | $h$ |
| 4) | $s_1$ | $b$ | $s_2$ | $c$ | $r$ |
| 5) | $s_1$ | $\|$ | $s_1$ | $\|$ | $r$ |
| 6) | $s_1$ | $c$ | $s_1$ | $c$ | $r$ |
| 7) | $s_2$ | $b$ | $s_3$ | $c$ | $l$ |
| 8) | $s_2$ | $\|$ | $s_2$ | $\|$ | $h$ |
| 9) | $s_2$ | $c$ | $s_2$ | $c$ | $h$ |
| 10) | $s_3$ | $b$ | $s_0$ | $b$ | $r$ |
| 11) | $s_3$ | $\|$ | $s_3$ | $\|$ | $l$ |
| 12) | $s_3$ | $c$ | $s_3$ | $c$ | $L$ |

**Tabelle 4**

$h_{t_4} : \{|\}^* \rightarrow \{b, |, c\}^*$

$h_{t_4}(|^n) = c^{2n}$ , $n \in \hat{u}_0$

Haltebereich: $\{|\}^*$

Wertebereich: $\{c^{2k} | k \in \hat{u}_0\}$

Beispiel: 2 Striche auf dem Band

$$\| \overset{2}{\rightarrow} b \,|\, \overset{5}{\rightarrow} b \,|\, \overset{4}{\rightarrow} b \,|\, c \overset{7}{\rightarrow} b \,|\, cc \overset{12}{\rightarrow} b \,|\, cc \overset{11}{\rightarrow} b \,|\, cc$$
$$\overset{10}{\rightarrow} b \,|\, cc \overset{2}{\rightarrow} bbcc \overset{6}{\rightarrow} bbcc \overset{6}{\rightarrow} bbcc \overset{4}{\rightarrow} bbccc$$
$$\overset{7}{\rightarrow} bbcccc \overset{12}{\rightarrow} bbcccc \overset{12}{\rightarrow} bbcccc \overset{12}{\rightarrow} bbcccc$$
$$\overset{10}{\rightarrow} bbcccc \overset{3}{\rightarrow} bbcccc$$

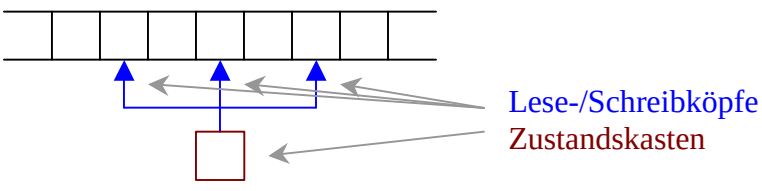
*Man beobachtet:* Der Wertebereich von $t_4$ „entspricht" dem Haltebereich von $t_3$.
(Man müsste noch die Striche von $t_3$ in *c*'s umwandeln.)
Somit kann man diese beiden Turing-Maschinen mit einander verknüpfen.

$$t_3 \circ t_4 : h_{t_3 \circ t_4}(|^n) = c^{2n+1}$$

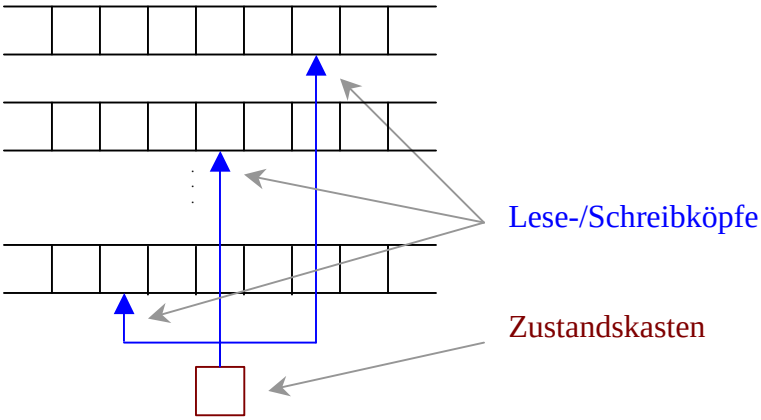
### 6.1.10 Varianten von Turing-Maschinen

Man kann zeigen, dass die Varianten der Turing-Maschinen nicht an Mächtigkeit gewinnen. Diese können alle durch die hier vorgestellte Turing-Maschine simuliert werden.

a) *k-Kopf-Turing-Maschine* (1 Band)

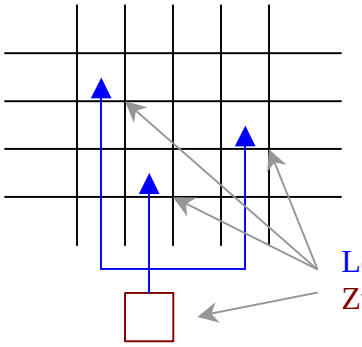


*b) k-Band-Turing-Maschine mit k Lese-/Schreibköpfe*




*c) k-dimensionale Turing-Maschine mit m Köpfe*

z.B.: $k = 2$
$m = 3$



Bemerkung: Man kann in alle Richtungen mit den LSK sich bewegen.



d) Turing-Maschinen mit Endzuständen statt der Aktion „*halt*“.

### 6.1.11 Universelle Turing-Maschinen*

An den wenigen Beispielen kann man schon erkennen, wie mächtig und flexibel Turing-Maschinen sind. Ein Problem gibt es jedoch, da sie nur eine ganz bestimmte Funktion (durch vorgegebenes Programm) berechnen kann. Diesen vermeintlichen Mangel kann man durch Konstruktion einer *universellen Turing-Maschine* beheben. Diese kann als Eingabe das Programm einer beliebig anderen Turing-Maschine annehmen und diese simulieren.

* nicht aus der Vorlesung

## 6.2 Registermaschinen

### 6.2.1 Warum Registermaschinen?

Ein Turing-Maschinen-Algorithmus, der die Addition zweier Unärzahlen beschreibt, lässt sich nicht so auf andere Berechnungsmodelle übertragen. Man führt nun (unter anderem deswegen) sogenannte *Registermaschinen* ein. Diese ähneln sehr konkrete Rechner im abstrakten Sinn. Sie hat eine endliche zahl von Registern, von denen jedes eine beliebig große natürliche Zahl speichern kann. Die Registermaschine rechnet nur mit natürlichen Zahlen, welches jedoch keine Einschränkung bedeutet. Computer rechnen genau genommen nur auf Basis von 0 und 1.

### 6.2.2 Beschreibung der Registermaschine

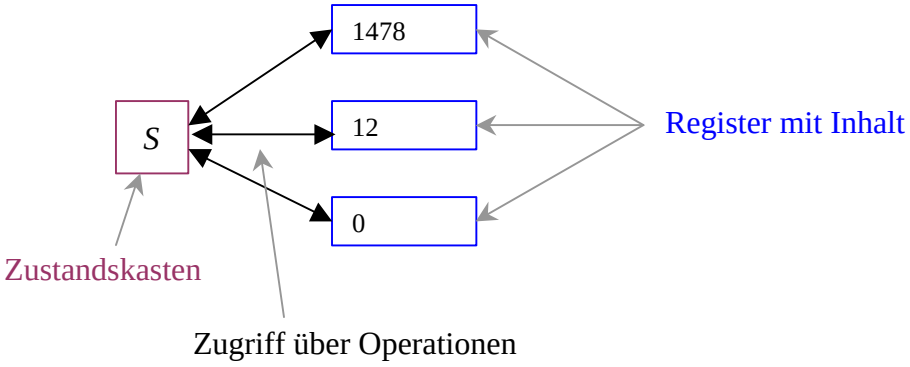
Wie bereits erwähnt verfügt die *Registermaschine* über eine *feste endliche Anzahl* (*k*) von Speicherzellen (Register). Jedes *Register* hat einen *unendlich großen Wertebereich* $\hat{u}_0$. Auf einen Register können drei verschiedene Operationen angewandt werden: Erhöhen des Inhalts um 1, Vermindern des Inhalts um 1 (, wobei jedoch die 0 nicht unterschritten werden darf) und die Abfrage des Inhalts auf 0.

### 6.2.3 Definition F:

Eine *k-Registermaschine* $r$ beschreibt man durch ein 5-Tupel $r = (S, k, d, s_0, F)$, wobei:

- $S$ endliche nichtleere Zustandsmenge
- $s_0 \in S$ Anfangszustand
- $k \in \hat{u}$ Registeranzahl
- $F \subseteq S$ Endzustandsmenge
- $d : (S \setminus F) \times \{0,1\}^k \rightarrow S \times \{-1,0,+1\}^k$
  - $\{0,1\}$ ist für den Nulltest: Wenn der Registerinhalt 0 ist, dann wird 0 zurückgegeben, sonst 1.
  - $\{-1,0,+1\}$ sind die Registeroperationen, wie oben beschrieben

Beispiel: 3-Registermaschine



### 6.2.4 Definition G:

Sei $r = (S, k, d, s_0, F)$ eine *Registermaschine*.

- $K_r = S \times \hat{u}_0^k$ heißt *Menge der Konfigurationen* von $r$.

- *Anfangskonfiguration*:
  $a : \hat{u}_0 \to K_r$ ist definiert durch $a(n) = (s_0, (n,0,...,0))$

*Start:* Im ersten Register steht der Startwert $n \in \hat{u}_0$ und alle anderen Register haben den Wert 0 und die Maschine befindet sich im Anfangszustand $s_0$.

- *Zustandsüberführungskonfiguration* (Schritt ausführen):

  $\hat{d} : K_r \to K_r$ mit $\hat{d}(s,(i_1,...,i_k)) = (s',(i_1',...,i_k')) :\Leftrightarrow$
  $d(s,(sign(i_1),...,sign(i_k))) = (s',(j_1',...,j_k'))$ mit $j_i \in \{+1,0,-1\}$

  $sign(i_l)$ ist die Vorzeichenfunktion. Sie liefert 0, wenn $i_l=0$ und 1, wenn $i_l>0$. ( $i_l<0$ ist nicht möglich. )

  $$i_l' = \begin{cases} i_l + j_j & ,sonst \\ 0 & ,falls\ i_l = 0 \wedge j_l = -1 \end{cases}$$

  Wendet die Registeroperationen an und verhindert das die 0 unterschritten wird.

*Arbeitsschritt:* Es wird die aktuelle Registerbelegung durch eine neue ersetzt, wobei auf den Registern nur die Operationen $\{+1,0,-1\}$ angewandt werden dürfen. Die Maschine geht in de Zustand *s'* über.

- *Ausgabefunktion* (Ergebnis):
  $w : K_r \to \hat{u}_0$ mit $w(s,(i_1,...,i_k)) = i_1$ , *falls* $s \in F$

*Ergebnis:* Die Registermaschine *hält*, wenn sie in einen Endzustand gelangt ist. Das Ergebnis ist dann der Wert der im *ersten* Register steht.

- Die von $r$ berechnete Funktion $h_r : \hat{u}_0 \to \hat{u}_0$ ist definiert durch

$$h_r(n) = \begin{cases} w(\hat{d}^m(a(n))) & ,falls\ m = Min\{j \mid \hat{d}^j(a(n)) = (s,(i_1,...,i_k))\ mit\ s \in F\ existiert\} \\ \perp & ,sonst \end{cases}$$

Diese Funktion ist nur dann definiert, wenn die Registermaschine *hält*. Dann hat sie ein Endzustand erreicht und die Berechnung ist zuende.

### 6.2.5 Definition H:

Sei $r = (S, k, d, s_0, F)$ eine *Registermaschine*.

a) Eine Abbildung $h : \hat{u}_0 \to \hat{u}_0$ heißt *RM$_k$-berechenbar*, wenn es eine Registermaschine $r$ mit *k-Registern* gibt, so dass $h_r = h$.

b) Kasse $\mathfrak{R}_{\mathbf{m_k}} = \{h \mid h \; RM_k \text{ - berechenbar ist}\}$

c) $\mathfrak{R}_{\mathbf{m}} = \bigcup_{k \geq 1} \mathfrak{R}_{\mathbf{m_k}}$ sei die Klasse aller *RM$_k$-berechenbaren* Funktionen

d) Eine Menge $M \subseteq \hat{u}_0$ heißt *Haltebereich* / Definitionsbereich einer Registermaschine $r$ mit *k-Registern*, so dass
$M = \{n \in \hat{u}_0 \mid h_r(n) \text{ definiert ist}\}$.

e) Eine Menge $M \subseteq \hat{u}_0$ heißt *Ergebnisbereich* / Wertebereich einer Registermaschine $r$ mit *k-Registern* mit
$M = \{m \in \hat{u}_0 \mid \exists n \in \hat{u}_0 \mid h_r(n) = m\}$.

### 6.2.6 Beispiele

Im folgenden nun ein paar Beispiele zu *Registermaschinen*.

**Beispiel 1:** $r_1 = (S_1 = \{s_0, s_1\}, k = 1, d_1, s_0, F_1 = \{s_1\})$ mit der Zustandsübergangsfunktion $d : (S_1 \setminus F_1) \times \{0,1\}^k \to S_1 \times \{-1,0,+1\}^k$ (siehe Tabelle 5).

Diese Registermaschine zählt das eine Register bis auf 0 herunter, addiert dann die 1 und landet im Endzustand $s_1$.

| $S_1$ | *Tests* | $S_1$ | *Operationen* |
|---|---|---|---|
| $s_0$ | 0 | $s_1$ | +1 |
| $s_0$ | 1 | $s_0$ | -1 |
| $s_1$ | 0 | - | - |
| $s_1$ | 1 | - | - |

- ≙ beliebig

**Tabelle 5**

*RM$_k$-berechenbare* Funktionen:

$h_{r_1}(n) = 1$ mit $n \in \hat{u}_0$

**Beispiel 2:** $r_2 = (S_2 = \{s_0, s_1, s_2, s_3, s_4\}, k = 2, d_2, s_0, F_2 = \{s_4\})$ mit der Zustandsübergangsfunktion $d : (S_2 \setminus F_2) \times \{0,1\}^k \rightarrow S_2 \times \{-1,0,+1\}^k$ (siehe Tabelle 6).

Diese Registermaschine vervierfacht den Wert, der im ersten Register steht, benutzt dazu ein Hilfsregister und landet im Endzustand $s_4$.

| $S_2$ | *Tests* | | $S_2$ | *Operationen* | |
|---|---|---|---|---|---|
| $s_0$ | 1 | - | $s_1$ | -1 | +1 |
| $s_0$ | 0 | 0 | $s_4$ | 0 | 0 |
| $s_0$ | 0 | 1 | $s_2$ | 0 | 0 |
| $s_1$ | - | - | $s_0$ | 0 | +1 |
| $s_2$ | - | 1 | $s_3$ | +1 | -1 |
| $s_2$ | - | 0 | $s_4$ | 0 | 0 |
| $s_3$ | - | - | $s_2$ | +1 | 0 |
| $s_4$ | - | - | - | - | - |

- ≙ beliebig

**Tabelle 6**

*RM$_k$-berechenbare* Funktionen:

$h_{r_2}(n) = 4n$ mit $n \in \hat{u}_0$

Insgesamt ergibt sich dann folgende Arbeitsweise:

$$(s_0, \begin{array}{|c|}\hline n \\ \hline 0 \\ \hline \end{array}) \rightarrow^* (s_2, \begin{array}{|c|}\hline 0 \\ \hline 2n \\ \hline \end{array}) \rightarrow^* (s_4, \begin{array}{|c|}\hline 4n \\ \hline 0 \\ \hline \end{array})$$

Beispiel: $n = 2$

$$\begin{array}{|c|}\hline 2 \\ \hline 0 \\ \hline \end{array} \overset{1}{\rightarrow} \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline \end{array} \overset{4}{\rightarrow} \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline \end{array} \overset{1}{\rightarrow} \begin{array}{|c|}\hline 0 \\ \hline 3 \\ \hline \end{array} \overset{4}{\rightarrow} \begin{array}{|c|}\hline 0 \\ \hline 4 \\ \hline \end{array} \overset{3}{\rightarrow} \begin{array}{|c|}\hline 0 \\ \hline 4 \\ \hline \end{array} \overset{5}{\rightarrow} \begin{array}{|c|}\hline 1 \\ \hline 3 \\ \hline \end{array} \overset{7}{\rightarrow} \begin{array}{|c|}\hline 2 \\ \hline 3 \\ \hline \end{array} \overset{5}{\rightarrow} \begin{array}{|c|}\hline 3 \\ \hline 2 \\ \hline \end{array} \overset{7}{\rightarrow} \begin{array}{|c|}\hline 4 \\ \hline 2 \\ \hline \end{array}$$

$$\overset{5}{\rightarrow} \begin{array}{|c|}\hline 5 \\ \hline 1 \\ \hline \end{array} \overset{7}{\rightarrow} \begin{array}{|c|}\hline 6 \\ \hline 1 \\ \hline \end{array} \overset{5}{\rightarrow} \begin{array}{|c|}\hline 7 \\ \hline 0 \\ \hline \end{array} \overset{7}{\rightarrow} \begin{array}{|c|}\hline 8 \\ \hline 0 \\ \hline \end{array} \overset{6}{\rightarrow} \begin{array}{|c|}\hline 8 \\ \hline 0 \\ \hline \end{array} \overset{8}{\rightarrow} \begin{array}{|c|}\hline 8 \\ \hline 0 \\ \hline \end{array}$$

**Beispiel 3:** Soll eine Registermaschine z.B. die Addition berechnen, so will man folgende *RM$_k$-Berechenbarkeitsfunktion* $h : \hat{u}_0^r \rightarrow \hat{u}_0^s$. Dazu muss man aber noch die Eingabekonfiguration *a* und die Ausgabekonfiguration *w* etwas modifizieren.

### 6.2.7 RAM-random access machine*

RAM ist eine verallgemeinerte Registermaschine, die einen wahlfreien Zugriff auf die Register und einen größeren Befehlsvorrat besitzt. Sie besteht aus einer zentralen Recheneinheit, einem Speicher, einem Eingabeband, einem Ausgabeband und einem Programm. Innerhalb der Komplexitätstheorie kann man mit ihnen Laufzeiten von Algorithmen bestimmen.

* nicht aus der Vorlesung

## Zusammenfassung

Turing-Maschinen haben einen endlichen Zustandsspeicher und ein unendliches Band auf das sie lesen, schreiben können, sowie ein Bandfeld nach links oder rechts bewegen können. Turing-Maschinen können nicht nur Sprachen erkennen, sondern besonders Funktionen berechnen. Es gibt viele Varianten von Turing-Maschinen, die die gleiche Mächtigkeit aber besitzen. Außerdem kann man universelle Turing-Maschinen bauen.

Registermaschinen sind leicht abstrahierte Rechner, die endlich viele Register besitzen, die eine natürliche Zahl aufnehmen können. Mit ihnen können Funktionen berechnet werden. Außerdem kann man mit ihnen auch Programme einer rudimentären imperativen Programmiersprache ausführen (nächste Vorlesungen).

Ergebnis: Turing-Maschinen und Registermaschinen haben weit an Mächtigkeit gegenüber Automaten, Push-Down-Akzeptoren, (bisher vorgestellten) gewonnen.

## Ausblick auf die nächste Vorlesung (das nächste Skript)

Wir werden im folgenden zeigen, dass Registermaschinen und Turing-Maschinen äquivalent zueinander sind. Die Beweisidee ist folgende: Man simuliert eine Turing-Maschine durch eine Registermaschine und umgekehrt, indem man jedes Register durch ein einseitig unendliches Band simuliert.

## Literaturverzeichnis

Erk, Katrin und Lutz Priese: Theoretische Informatik / Eine umfassende Einführung, Springer-Verlag 2000, erste Auflage

Schöning, Uwe: Theoretische Informatik – kurzgefasst, Spektrum Akademischer Verlag, 1995, zweite Auflage

Klaus, Volker und Schwill, Andreas: Schülerduden Informatik, Dudenverlag, 1997, dritte neu bearbeitete Auflage