Vorlesungsmitschrift:
Theoretische Informatik II

Christian Franz

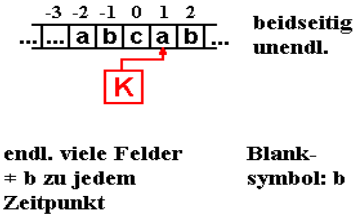
01.12.2000

Inhaltsverzeichnis

Turingmaschinen

Um Chomsky-Sprachen vom Typ 0 beschreiben zu können, reichen die Kellerautomaten nicht mehr aus. Es wurde also nach einem neuen Typ von Automat für diese Sprachklasse gesucht. Ein Automatenmodell was diese Aufgabe bewältigt schlug Alan M. Turing[1] vor.

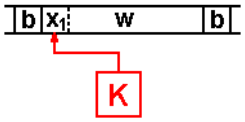
Anschaulich beschrieben besteht eine Turingmaschine aus einem beidseitig, unendlichem Band, das in Felder eingeteilt ist. Jedes Feld kann ein einzelnes Zeichen des Alphabets (Γ) der Maschine enthalten. Auf dem Band kann sich ein Schreib-Lesekopf (LSK) bewegen. Nur solche Zeichen, auf denen sich dieser Kopf befindet, können im momentanen Rechenschritt verändert werden. Außerdem kann dieser Schreib-Lesekopf in einem Rechenschritt nur um eine Position nach links oder rechts bewegt werden.



Arbeitsweise der Turingmaschine

Im folgenden Abschnitt soll nun die Arbeitsweise einer Turingmaschine kurz skizziert werden:

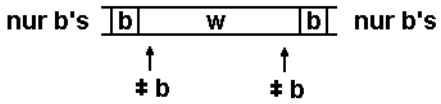
a) **Initialisierung:** $w \in X^*, w = x_1, \ldots, x_n \qquad x_i \in X \subset \Gamma$



[1] Alan M. Turing, 1912–1954, englischer Mathematiker, Kryptoanalytiker und Computerkonstrukteur vor. Seine 1937 erschienene Arbeit „On computable numbers, with an application to the Entscheidungsproblem“ hat die Berechenbarkeitstheorie, und damit die Theorie der Informatik, begründet.

b) **Schritt ausführen:**
(Zeichen unter LSK, akt. Zustand) $\overset{\delta}{\rightarrow}$ (neuer Zustand, neues Zeichen unter LSK)
LSK-Bewegung: l (links), r (rechts), h (stop)
$\delta(s,a) = (s',a',P) \qquad s,s' \in S;\ \ a,a' \in \Gamma;\ \ P \in \{l,r,h\}$

c) **Berechnungsende:**



Definition A

Die oben gestellten Anforderungen an eine Turingmaschine werden nun formal spezifiziert:

Ein 6-Tupel $\tau = (S, X, \Gamma, \delta, s_0, b)$ heißt Turingmaschine, wenn gilt:

- S ist eine endl. nichtleere Menge von Zuständen,
- $s_0 \in S$ ist der Anfangszustand,
- $X \in \Gamma$ ist das Eingabealphabet,
- $b \in \Gamma \setminus X$ ist das Blanksymbol und
- $\delta : S \times \Gamma \to S \times \Gamma \times \{l,r,h\}$
  ist eine partielle Funktion.

Definition B

Um den aktuellen Zustand in dem sich eine Turingmaschine befindet besser beschreiben zu können, wird der Begriff der Konfiguration eingeführt:

Ein Tripel (s, f, i) heißt **Konfiguration** der Turingmaschine, wenn gilt:

(i) $s \in S$ ist der aktuelle Zustand.

(ii) $f : \mathbb{Z} \to \Gamma$ ist der Bandinhalt. Dabei ist $f(j)$ der Inhalt der j-ten Zelle und es existieren nur endlich viele j mit $f(j) \neq b$.

(iii) $i \in \mathbb{Z}$ ist die Position des LSK.

Die Menge aller Konfigurationen von τ wird mit K_τ bezeichnet.

Definition C

Die Arbeitsweise einer Turingmaschine ist wie folgt definiert:

(1) **Anfangskonfiguration:** Sei $w \in X^*$ mit $w = w_0, \ldots, w_k \qquad w_i \in X$.
Dann gilt:
$\alpha : X^* \to K_\tau$ mit $\alpha(w) = (s_0, f_w, \delta)$ $\qquad f_w$:Bandinschrift

$$f_w(j) = \begin{cases} w_j, 0 \leq j \leq k \\ \text{b, sonst} \end{cases}$$

(2) **Schritt ausführen:** $\hat{\delta} : K_\tau \to K_\tau$ mit $\hat{\delta}(s, f, i) = (s', f', i')$
$:\Rightarrow f(i) = a \in \Gamma \quad \delta(s, a) = (s', a', P)$

$$f'(j) = \begin{cases} f(j), \text{ falls } j \neq i \\ a', \text{ falls j=i} \end{cases}$$

$$i' = \begin{cases} i + 1, \text{ falls P=r} \\ i - 1 \text{ , falls P=l} \\ i \text{ , falls P=h} \end{cases}$$

3) **Ergebnis:** $w : K_\tau \to \Gamma^*$ mit
$w(s, f, i) = \epsilon$, falls $f(j) = b$ für alle $j \in \mathbb{Z}$
$w(s, f, i) = v_0 v_1 \ldots v_l$, falls es ein $k \in \mathbb{Z}$ gibt, mit:

- $f(j) = b$, für alle $j < k, f(k) \neq b$
- $f(j) = b$, für alle $j > k + l, f(k + l) \neq b$
- $f(j) = v_{j-k}$, für $k \leq j \leq k + l$

Es gibt genau ein k!

Definition D

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$ eine Turingmaschine.
Die von τ berechnete Funktion $h_\tau : X^* \to \Gamma^*$ ist definiert durch:

$$h_\tau(w) = \begin{cases} w(\hat{\delta}^{m+1}(\alpha(w))), \text{ falls } m = Min\{j \mid \hat{\delta}^j(\alpha(w)) = (s, f, i) \text{ und} \\ \exists s', a' : \delta(s, f(i)) = (s', a', h)\} \\ \perp, \text{ sonst (Endlosschleife)} \end{cases}$$

Nachdem in der letzten Woche der Begriff der Turingmaschine eingeführt wurde, soll nun der Begriff der Turing–Berechenbarkeit definiert werden:

Definition E

Seien A und B Alphabete.

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

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

c) τ sei die Klasse aller Turing-berechenbaren Funktionen (A, B beliebig)

d) Eine Menge $M \subseteq B^*$ heit Haltebereich bzw. Definitionsbereich einer Turingmaschine $\tau = (S, A, \Gamma, \delta, s_0, b)$, wenn
$M = \{w \in A^* \mid h_\tau(w) \quad def.\}$.

e) Eine Menge $M \subseteq B^*$ heißt Ergebnisbereich bzw. Wertebereich einer Turingmaschine $\tau = (S, A, \Gamma, \delta, s_0, b)$ mit $B \subseteq \Gamma$, wenn
$M = \{v \in B^* \mid \exists w \in A^* : h_\tau(w) = v\}$.

Beispiele

Im folgenden Abschnitt werden einige einfache Turingmaschinen vorgestellt. Dabei kann man mehrere Turingmaschinen in einer Einzelnen zusammenfassen, um so neue Maschinen zu erhalten, die „schwierigere“ Sprachen erkennen können.

1. $\tau_1 = (\{s_0\}, \{|\}, \{b, |\}, \delta_1, s_0, b)$ mit
$\delta(s_0, b) = (s_0, |, h)$ und $\delta(s_0, |) = (s_0, |, r)$

S	Γ	S	Γ	-
s_0	|	s_0	|	r
s_0	b	s_0	|	h

$h_{\tau_1} : \{|\}^* \to \{b, |\}^*$
$h_{\tau_1}(|^n) = |^{n+1}$ $\qquad n \in \mathbb{N}$
Haltebereich: $\{|\}^*$
Wertebereich: $\{|\}^+$

2. $\tau_2 = (\{s_0\}, \{|\}, \{b, |\}, \delta_2, s_0, b)$ (Nullfunktion)

S	Γ	S	Γ	-
s_0	b	s_0	b	h
s_0	|	s_0	b	r

$h_{\tau_2}(w) = \epsilon, w \in \{|\}^*$
Haltebereich: $\{|\}^*$
Wertebereich: $\{\epsilon\}$

3. $\tau_3 = (\{s_0, s_1\}, \{|\}, \{b, |\}, \delta_3, s_0, b)$ (Geradzahligkeitsmaschine)

S	Γ	S	Γ	-
s_0	b	s_1	|	h
s_0	|	s_1	|	r
s_1	b	s_1	b	r
s_1	|	s_0	|	r

$h_{\tau_3}(|^n) = \begin{cases} |^{n+1}, \text{ falls n durch 2 teilbar ist} \\ \text{undef. sonst} \end{cases}$

4. $\tau_4 = (\{s_0, s_1, s_2, s_3\}, \{|\}, \{|, c, b\}, \delta_4, s_0, b)$ (Dopplungsmaschine)

S	Γ	S	Γ	-
s_0	b	s_0	b	h
s_0	|	s_1	b	r
s_0	c	s_0	c	h
s_1	b	s_2	c	r
s_1	|	s_1	|	r
s_1	c	s_1	c	r
s_2	b	s_3	c	l
s_2	|	s_2	|	h
s_2	c	s_2	c	h
s_3	b	s_0	b	r
s_3	|	s_3	|	l
s_3	c	s_3	c	l

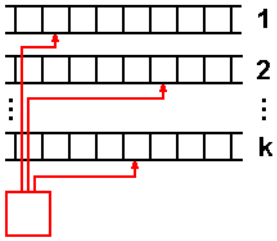
$h_{\tau_4}(|^n) = c^{2n}$ $n \in \mathbb{N}_0$

Beobachte: Der Wertebereich von τ_4 „entspricht“ dem Haltebereich von τ_3
Setze $\tau_3 \circ \tau_4 : h_{\tau_3 \circ \tau_4}(|^n) = c^{2n+1}$

Varianten von Turingmaschinen

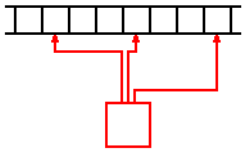
Im folgenden Abschnitt werden einige weitere Varianten von Turingmaschinen vorgestellt. Man kann jedoch zeigen, daß alle diese Varianten *nicht* leistungsfähiger sind, als eine Turingmaschine mit einem Band, so wie sie in Definition A festgelegt wurde. [2]

a) k-Band-Turingmaschine



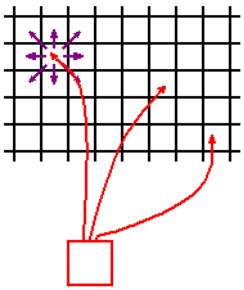
Fazit: Man gewinnt nichts an Leistungsfähigkeit, da man k Bänder durch ein Einzelnes simulieren kann.

b) k-Kopf-Turingmaschine



Ebenfalls kein Gewinn an Leistungsfähigkeit.

c) k-dimensionale Turingmaschine mit m Köpfen z.B.: k=2 und m=3



[2] Ein Beweis findet sich z.B. in [EP2000], S. 174ff.

Auch hier kann man zeigen, daß eine Simulation durch eine 1-Band Turingmaschine möglich ist.

d) Turingmaschine mit Endzuständen statt der Aktion „halt“

6.2. Registermaschinen

In diesem Abschnitt geht es um Registermaschinen, die realen Rechnern sehr ähneln (bis auf die Tatsache, daß sie beliebig viel Speicher zur Verfügung haben). Registermaschinen haben eine endliche Anzahl von Registern, von denen jedes eine beliebig große natürliche Zahl speichern kann.
Man kann zeigen, daß Registermaschinen und Turingmaschinen äquivalent bezüglich ihrer Leistungsfähigkeit sind.

TM: unendlich viel Speicher/endlicher Wertebereich der Zellen

RM: endlicher Speicher/unendlicher Wertebereich $\mathbb{N}_0$

Definition F

Eine k-Registermaschine ρ beschreibt man durch ein
5-Tupel $\rho = (S, k, \delta, s_0, F)$, wobei

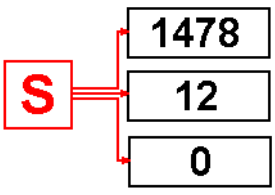
i) S eine endliche Menge ($\neq \emptyset$) von Zuständen,

ii) $s_0 \in S$ der Anfangszustand,

iii) $k \in \mathbb{N}$ die Anzahl der Register,

iv) $F \subseteq S$ die Menge der Endzustände ist.

v) $\delta : (S \setminus F) \times \underbrace{\{0,1\}^k}_{Nulltest} \rightarrow \underbrace{S \times \{-1,0,1\}^k}_{Registeroperationen}$



3-Registermaschine

Definition G

Genau wie bei der Turingmaschine, kann man auch bei der Registermaschine eine Konfiguration angeben, die den aktuellen Zustand der Maschine widerspiegelt.

(i) $K_\rho = S \times \mathbb{N}_0{}^k$ heißt Menge der Konfigurationen von ρ.

(ii) $\alpha : \mathbb{N}_0 \to K_\rho$ ist definiert durch $\alpha(n) = (s_0, (n, 0, \ldots, 0))$

(iii) $\hat{\delta} : K_\rho \to K_\rho$ mit $\hat{\delta}(s, (i_1, \ldots, i_k)) = (s', (i'_1, \ldots, i'_k))$
$:\Leftrightarrow$ Es gilt $\delta(s, (sign(i_1), \ldots, sign(i_k))) = (s', (j_1, \ldots, j_k))$

$$i'_l = \begin{cases} i_l + j_l, \text{ sonst} \\ 0, \text{ falls } i_l = 0, j_l = -1 [\text{modifizierte Subtraktion } \dot{-}\ ] \end{cases}$$

Wende Registeroperatoren +1, -1, 0 auf die Register an.

(iv) $\omega : K_\rho \to \mathbb{N}_0$ mit $\omega(s, (i_1, \ldots, i_k)) = i_1$, falls $s \in F$

(v) $h_\rho : \mathbb{N}_0 \to \mathbb{N}_0$ ist definiert durch

$$h_\rho(n) = \begin{cases} \omega(\hat{\delta}^m(\alpha(n))), \text{ falls} \\ m = min\{j \mid \hat{\delta}^j(\alpha(n)) = (s, \ldots) \quad \text{mit} \quad s \in F\} \quad \text{ex.} \\ \\ \text{undefiniert sonst} \end{cases}$$

ist die berechnete Funktion von ρ.

Definition H

(i) Eine Abbildung $h : \mathbb{N}_0 \to \mathbb{N}_0$ heißt RM_k-berechenbar, wenn es eine Registermaschine mit k Registern gibt, so daß $h = h_\rho$ gilt.

(ii) $\mathcal{R}m_k = \{h \mid h \text{ ist } RM_k\text{-berechenbar}\}$

(iii) $\mathcal{R}m = \bigcup_{k \geq 1} \mathcal{R}m_k$

(iv) $M \in \mathbb{N}_0$ heißt Haltebereich einer Registermaschine, wenn es ein k und eine RM ρ mit k Registern gibt, so daß
$M = \{n \in \mathbb{N}_0 \mid h_\rho(n) \text{ ist def. }\}$

(v) $M \subseteq \mathbb{N}_0$ heißt Werte– bzw. Ergebnisbereich einer Registermaschine, wenn es eine RM ρ mit k Registern gibt, so daß
$M = \{m \in \mathbb{N}_0 \mid \exists n \in \mathbb{N}_0 : h_\rho(n) = m\}$ gilt.

Beispiele

Es werden nun einige Beispiele für sehr einfache Registermaschinen vorgestellt:

1.) $\rho_1 = (\{s_0, s_1\}, 1, \delta, s_0, \{s_1\})$

S	Test	S	Operator
s_0	0	s_1	+1
s_0	1	s_0	-1
s_1	0	-	-
s_1	1	-	-

Zähle Register auf 0 und addiere 1: $h_{\rho_1}(n) = 1$

2.) $\rho_2 = (\{s_0, s_1, s_2, s_3, s_4\}, 2, \delta, s_0, \{s_4\})$

S	Test	S	Operator
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

„-“ entspricht beliebig

Arbeitsweise: $\left(s_0, \begin{array}{|c|}\hline \text{n} \\ \hline 0 \\ \hline \end{array}\right) \rightsquigarrow^* \left(s_2, \begin{array}{|c|}\hline 0 \\ \hline \text{2n} \\ \hline \end{array}\right) \rightsquigarrow^* \left(s_4, \begin{array}{|c|}\hline \text{4n} \\ \hline 0 \\ \hline \end{array}\right)$

Offenbar: $h_{\rho_2}(n) = 4n$

3.) **Erweiterung:** Will man $h : {\mathbb{N}_0}^r \rightarrow {\mathbb{N}_0}^s$, so modifiziert man α und ω. So kann man auch die Addition berechnen.

Index

Literaturverzeichnis

[EP2000] Erk, K., Priese, L, „Theoretische Informatik – Eine umfassende Einführung“, Springer, 2000.

[Ko1997] Kozen, D. C., „Automata and Computability“, New York, Springer, 1997.

[Sch1999] Schöning, U., „Theoretische Informatik – kurzgefaßt“, Berlin, Spektrum Akademischer Verlag, 1999.