Vorlesung

Grundlagen der Theoretischen Informatik / Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik

UNIVERSITÄT KOBLENZ · LANDAU

Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)
Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– Bernhard Beckert, April 2007

Immer mächtigere Automaten

Erinnerung: Endliche Automaten

- akzeptieren reguläre Sprachen
- Einziger Speicher: der **Zustand** (endlich).
- Die Zustandsmenge kann groß sein, ist aber endlich.
- Das Eingabewort wird nur einmal gelesen, von links nach rechts.

Immer mächtigere Automaten

Erinnerung: Pushdown-Automaten

- akzeptieren kontextfreie Sprachen
- Erster Speicher: der Zustand (endlich)
- Zweiter Speicher: der **Keller**
  (unbeschränkte Größe, beschränkte Zugriffsart)
- Das Eingabewort wird nur einmal gelesen, von links nach rechts.

Immer mächtigere Automaten

Ausblick: Turing-Maschinen

- akzeptieren Sprachen **vom Typ 0**.
- Erster Speicher: der Zustand (endlich)
- Zweiter Speicher: Band
  (unbeschränkte Größe, **Zugriff an beliebiger Stelle**
- Turing-Maschine hat einen Schreib-/Lesekopf, den sie über diesem Band in einem Rechenschritt um ein Feld nach rechts oder links bewegen kann.
- Das Eingabewort steht (am Anfang) auf dem Band.
  Die Maschine kann es **beliebig oft lesen**.

Turing-Maschine

Definition 9.1 (Turing-Maschine (DTM))

Eine **determinierte Turing-Maschine (DTM)** $\mathcal{M}$ ist ein Tupel

$$\mathcal{M} = (\ K, \Sigma, \delta, s\ )$$

Dabei ist

- K eine endliche Menge von Zuständen mit $h \notin K$,
  (h ist der **Haltezustand**)
- Σ ein Alphabet mit $L, R \notin \Sigma$, $\# \in \Sigma$,
- $\delta : K \times \Sigma \to (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$ eine Übergangs**funktion**
- $s \in K$ ein Startzustand.

Anzahl der Zustände: $|K| - 1$
(Startzustand wird nicht mitgezählt).

Turing-Maschine

Arbeitsschritt einer Turing-Maschine

Übergang

$$\delta(q, a) = (q', x)$$

bedeutet:

In Abhängigkeit

- vom aktuellen Zustand $q \in K$
- von dem Zeichen $a \in \Sigma$, das unter dem Schreib-/Lesekopf steht

geschieht folgendes:

- entweder ein **Schritt nach links**, falls $x = L$ ist
- oder ein **Schritt nach rechts**, falls $x = R$ ist
- oder **das Zeichen** a, das momentan unter dem Schreib-/Lesekopf steht, wird **durch** $b \in \Sigma$ **überschreiben**, falls $x = b \in \Sigma$
- der **Zustand** wird zu $q' \in K \cup \{h\}$ **geändert**,

Turing-Maschine

Leerzeichen

Das spezielle Zeichen # (*blank*) ist das Leerzeichen.

Es ist nie Teil des Eingabeworts; man kann es u.a. dazu benutzen, Wörter voneinander abzugrenzen.

Begrenzung des Bandes

Das Band einer DTM ist **einseitig unbeschränkt**:

- Nach rechts ist es unendlich lang.
- Nach links hat es ein Ende.
- Wenn eine DTM versucht, das Ende zu überschreiten, bleibt sie „hängen“.

  In diesem Fall hält sie nicht.

Anfangskonfiguration

- Ganz links auf dem Band steht ein Blank
- Direkt rechts davon steht das Eingabewort
- Wenn eine DTM mehrere Eingabewörter hintereinander bekommt, sind sie durch Blanks getrennt.
- Rechts vom letzten Eingabewort stehen nur noch Blanks.
- Der Schreib-/Lesekopf der DTM steht auf dem Blank direkt rechts neben dem (letzten) Eingabewort.

Merke

Das Band enthält immer nur endlich viele Symbole, die keine Blanks sind.

Beispiel 9.2 ($\mathcal{R}(a)$: *a*’s durch *b*’s ersetzen)

Die folgende Turing-Maschine $\mathcal{R}(a)$ erwartet *ein* Eingabewort.
Sie liest es von rechts nach links einmal durch und macht dabei jedes *a* zu einem *b*.

Es ist

$$\mathcal{R}(a) = (\ \{q_0, q_1\}, \{a, b, \#\}, \delta, q_0\ )$$

mit folgender δ-Funktion:

$$\begin{array}{ll} q_0, \# \mapsto q_1, L & q_1, \# \mapsto h, \# \\ q_0, a \mapsto q_0, a & q_1, a \mapsto q_1, b \\ q_0, b \mapsto q_0, b & q_1, b \mapsto q_1, L \end{array}$$

Beispiel 9.3 ($\mathcal{L}_\#$)

Die folgende Turing-Maschine $\mathcal{L}_\#$ läuft zum ersten Blank links von der momentanen Position.

Es ist $\mathcal{L}_\# = (\ \{q_0, q_1\}, \{a, b, \#\}, \delta, q_0\ )$ mit folgender δ-Funktion:

$$\begin{array}{ll} q_0, \# \mapsto q_1, L & q_1, \# \mapsto h, \# \\ q_0, a \mapsto q_1, L & q_1, a \mapsto q_1, L \\ q_0, b \mapsto q_1, L & q_1, b \mapsto q_1, L \end{array}$$

q_0: Anfangsposition
q_1: Anfangsposition verlassen

Turing-Maschine

Beispiel 9.4 (Copy)

Die folgende DTM $\mathcal{C}$ erhält als Eingabe einen String Einsen.
Dieser String wird kopiert:
Falls n Einsen auf dem Band stehen, stehen nach Ausführung von $\mathcal{C}$ $2n$ Einsen auf dem Band stehen, (getrennt durch ein Blank #).

state	#	1	c
q_0	$\langle q_1, c\rangle$	–	–
q_1	$\langle q_2, R\rangle$	$\langle q_1, L\rangle$	$\langle q_1, L\rangle$
q_2	–	$\langle q_3, \#\rangle$	$\langle q_7, \#\rangle$
q_3	$\langle q_4, R\rangle$	–	–
q_4	$\langle q_5, 1\rangle$	$\langle q_4, R\rangle$	$\langle q_4, R\rangle$
q_5	$\langle q_6, 1\rangle$	$\langle q_5, L\rangle$	$\langle q_5, L\rangle$
q_6	–	$\langle q_2, R\rangle$	–
q_7	$\langle q_8, R\rangle$	–	–
q_8	$\langle h, \#\rangle$	$\langle q_8, R\rangle$	–

Turing-Maschine

Übergangsfunktion δ nicht überall definiert

Wir erlauben ab jetzt auch, daß δ nicht überall definiert ist.

Falls die DTM dann in einen solchen nichtdefinierten Zustand kommt, sagen wir **die DTM hängt**. Sie **hält** also nicht.

Dies wird z.T. in der Literatur anders gehandhabt.

Turing-Maschine

Beispiel 9.5 (Print n)

Für jedes $n \in \mathbb{N}$ konstruieren wir eine Maschine,
die genau n Einsen auf das leere Band schreibt
(**mit möglichst wenig Zuständen**):

1. schreibe $\lfloor \frac{n}{2} \rfloor$ viele Einsen auf das Band
   (höchstens $\lfloor \frac{n}{2} \rfloor$ Zustände)
2. kopiere diesen String
   (8 Zustände)
3. ersetze das trennende # durch eine 1
4. falls n gerade ist, ersetzen die letzte 1 durch # (2 neue Zustände)

Insgesamt können wir n Einsen mit höchstens $\lfloor \frac{n}{2} \rfloor + 10$ Zuständen konstruieren

Turing-Maschine

Begriff der Konfigurationen

- **Konfiguration** beschreibt die *komplette* aktuelle Situation der Maschine in einer Rechnung.
- Eine **Rechnung** ist eine Folge von Konfigurationen,
  wobei immer von einer Konfiguration zu einer Nachfolgekonfiguration übergegangen wird.

Konfiguration einer DTM

Besteht aus 4 Elementen:

- das aktuellen Zustand q,
- das Wort w links vom Schreib-/Lesekopf,
- das Zeichen a, auf dem der Kopf gerade steht,
- das Wort u rechts von der aktuellen Kopfposition.

Konfigurationen sind endlich

- w enthält das Anfangsstück des Bandes vom linken Ende bis zur aktuellen Kopfposition.
- **Links ist das Band endlich!**
  $w = \varepsilon$ bedeutet, daß der Kopf ganz links steht
- u enthält den Bandinhalt rechts vom Schreib-/Lesekopf bis zum letzten Zeichen, das kein Blank ist.
- **Nach rechts ist das Band unendlich, aber es enthält nach rechts von einer bestimmten Bandposition an nur noch Blanks**.
  $u = \varepsilon$ bedeutet, daß rechts vom Schreib-/Lesekopf nur noch Blanks stehen.

Definition 9.6 (Konfiguration einer DTM)

Eine **Konfiguration** C einer DTM $\mathcal{M} = (K, \Sigma, \delta, s)$ ist ein Wort der Form $C = q, w\underline{a}u$. Dabei ist

- $q \in K \cup \{h\}$ der aktuelle Zustand,
- $w \in \Sigma^*$ der Bandinhalt links des Kopfes,
- $a \in \Sigma$ das Bandzeichen unter der Schreib-/Lesekopf.
  Notation: Die Position des Schreib-/Lesekopfes ist durch einen Unterstrich gekennzeichnet.
- $u \in \Sigma^*(\Sigma - \{\#\}) \cup \{\varepsilon\}$ der Bandinhalt rechts des Kopfes.

Definition 9.7 (Nachfolgekonfiguration)

Eine Konfiguration C_2 heißt **Nachfolgekonfiguration** von C_1, in Zeichen

$$C_1 \vdash_{\mathcal{M}} C_2$$

falls gilt:

- $C_i = q_i, w_i\underline{a_i}u_i$ für $i \in \{1, 2\}$, und
- es gibt einen Übergang $\delta(q_1, a_1) = (q_2, b)$ wie folgt:
  Fall 1: $b \in \Sigma$. Dann ist $w_1 = w_2,\ u_1 = u_2,\ a_2 = b$.
  Fall 2: $b = L$. Dann gilt für w_2 und a_2: $w_1 = w_2 a_2$.
  Für u_2 gilt: Wenn $a_1 = \#$ und $u_1 = \varepsilon$ ist, so ist $u_2 = \varepsilon$, sonst ist $u_2 = a_1 u_1$.
  Fall 3: $b = R$. Dann ist $w_2 = w_1 a_1$.
  Für a_2 und u_2 gilt: Wenn $u_1 = \varepsilon$ ist, dann ist $u_2 = \varepsilon$ und $a_2 = \#$, ansonsten ist $u_1 = a_2 u_2$.