SS 2005

Einführung in die Informatik IV

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2005SS/info4/index.html.de

30. Mai 2005

Definition 98

Eine Konfiguration einer Turingmaschine ist ein Tupel $(\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$.

Das Wort $w = \alpha\beta$ entspricht dem Inhalt des Bandes, wobei dieses rechts und links von w mit dem Leerzeichen $\Box$ gefüllt sei. Der Schreib-/Lesekopf befindet sich auf dem ersten Zeichen von $\beta\Box^\infty$.

Die Startkonfiguration der Turingmaschine bei Eingabe $x \in \Sigma^*$ entspricht der Konfiguration

$$(\epsilon, q_0, x)\,,$$

d.h. auf dem Band befindet sich genau die Eingabe $x \in \Sigma^*$, der Schreib-/Lesekopf befindet sich über dem ersten Zeichen der Eingabe und die Maschine startet im Zustand q_0.

Definition 98

Eine Konfiguration einer Turingmaschine ist ein Tupel $(\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$.

Das Wort $w = \alpha\beta$ entspricht dem Inhalt des Bandes, wobei dieses rechts und links von w mit dem Leerzeichen $\Box$ gefüllt sei. Der Schreib-/Lesekopf befindet sich auf dem ersten Zeichen von $\beta\Box^\infty$.

Die Startkonfiguration der Turingmaschine bei Eingabe $x \in \Sigma^*$ entspricht der Konfiguration

$$(\epsilon, q_0, x)\,,$$

d.h. auf dem Band befindet sich genau die Eingabe $x \in \Sigma^*$, der Schreib-/Lesekopf befindet sich über dem ersten Zeichen der Eingabe und die Maschine startet im Zustand q_0.

Definition 98

Eine Konfiguration einer Turingmaschine ist ein Tupel $(\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$.

Das Wort $w = \alpha\beta$ entspricht dem Inhalt des Bandes, wobei dieses rechts und links von w mit dem Leerzeichen $\square$ gefüllt sei. Der Schreib-/Lesekopf befindet sich auf dem ersten Zeichen von $\beta\square^\infty$.

Die Startkonfiguration der Turingmaschine bei Eingabe $x \in \Sigma^*$ entspricht der Konfiguration

$$(\epsilon, q_0, x)\,,$$

d.h. auf dem Band befindet sich genau die Eingabe $x \in \Sigma^*$, der Schreib-/Lesekopf befindet sich über dem ersten Zeichen der Eingabe und die Maschine startet im Zustand q_0.

Je nach aktuellem Bandinhalt und Richtung $d \in \{L, R, N\}$ ergibt sich bei Ausführung des Zustandsübergangs $\delta(q, \beta_1) = (q', c, d)$ folgende Änderung der Konfiguration:

$$
(\alpha_1 \cdots \alpha_n, q, \beta_1 \cdots \beta_m) \to
\begin{cases}
(\alpha_1 \cdots \alpha_n, q', c\beta_2 \cdots \beta_m) & \text{falls } d = N, \\
 & n \geq 0, m \geq 1 \\
(\epsilon, q', \Box c\beta_2 \cdots \beta_m) & \text{falls } d = L, \\
 & n = 0, m \geq 1 \\
(\alpha_1 \cdots \alpha_{n-1}, q', \alpha_n c\beta_2 \cdots \beta_m) & \text{falls } d = L, \\
 & n \geq 1, m \geq 1 \\
(\alpha_1 \cdots \alpha_n c, q', \Box) & \text{falls } d = R, \\
 & n \geq 0, m = 1 \\
(\alpha_1 \cdots \alpha_n c, q', \beta_2 \cdots \beta_m) & \text{falls } d = R, \\
 & n \geq 0, m \geq 2
\end{cases}
$$

Der Fall $m = 0$ wird mittels $\beta_1 = \Box$ abgedeckt.

Definition 99

Die von einer Turingmaschine M akzeptierte Sprache ist

$$L(M) = \{x \in \Sigma^*;\ (\epsilon, q_0, x) \rightarrow^* (\alpha, q, \beta) \text{ mit } q \in F, \alpha, \beta \in \Gamma^*\}$$

5.2 Linear beschränkte Automaten

Definition 100

Eine Turingmaschine heißt linear beschränkt (kurz: LBA), falls für alle $q \in Q$ gilt:

$$(q', c, d) \in \delta(q, \Box) \qquad \Longrightarrow \qquad c = \Box\,.$$

Ein Leerzeichen wird also nie durch ein anderes Zeichen überschrieben. Mit anderen Worten: Die Turingmaschine darf ausschliesslich die Positionen beschreiben, an denen zu Beginn die Eingabe x steht.

5.2 Linear beschränkte Automaten

Definition 100

Eine Turingmaschine heißt linear beschränkt (kurz: LBA), falls für alle $q \in Q$ gilt:

$$(q', c, d) \in \delta(q, \Box) \qquad \Longrightarrow \qquad c = \Box\,.$$

Ein Leerzeichen wird also nie durch ein anderes Zeichen überschrieben. Mit anderen Worten: Die Turingmaschine darf ausschliesslich die Positionen beschreiben, an denen zu Beginn die Eingabe x steht.

Satz 101

Die von linear beschränkten, nichtdeterministischen Turingmaschinen akzeptierten Sprachen sind genau die kontextsensitiven (also Chomsky-1) Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

Satz 101

Die von linear beschränkten, nichtdeterministischen Turingmaschinen akzeptierten Sprachen sind genau die kontextsensitiven (also Chomsky-1) Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

Satz 101

Die von linear beschränkten, nichtdeterministischen Turingmaschinen akzeptierten Sprachen sind genau die kontextsensitiven (also Chomsky-1) Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

„$\Longrightarrow$": Wir benutzen eine Menge von Nichtterminalsymbolen, die $Q \times \Sigma$ enthält, für die Grammatik. Die Grammatik erzeugt zunächst alle akzeptierenden Konfigurationen (der Form $\alpha q_f \beta$ mit $q_f \in F$), wobei q_f und β_1 zusammen als ein Zeichen codiert sind. Sie enthält weiterhin Regeln, die es gestatten, aus jeder Satzform, die eine Konfiguration darstellt, alle möglichen unmittelbaren Vorgängerkonfigurationen abzuleiten. Die zur Anfangskonfiguration (ϵ, q_0, w) gehörige Satzform ist damit ableitbar gdw der LBA das Wort w akzeptiert.

Satz 101

Die von linear beschränkten, nichtdeterministischen Turingmaschinen akzeptierten Sprachen sind genau die kontextsensitiven (also Chomsky-1) Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

„$\Longleftarrow$“: Wir simulieren mittels des LBA nichtdeterministisch und in Rückwärtsrichtung die möglichen Ableitungen der kontextsensitiven Grammatik und prüfen, ob wir die Satzform S erreichen können. Da die Grammatik längenmonoton ist, nimmt der vom LBA benötigte Platz nie zu. □

Beispiel 102

Die Sprache $L = \{a^m b^m c^m \mid m \in \mathbb{N}_0\}$ ist kontextsensitiv.

Satz 103

Das Wortproblem für LBAs bzw. für Chomsky-1-Grammatiken ist entscheidbar.

Beweis:

Siehe z.B. die Konstruktion zum vorhergehenden Satz bzw. Übungsaufgabe. □

Satz 103

Das Wortproblem für LBAs bzw. für Chomsky-1-Grammatiken ist entscheidbar.

Beweis:

Siehe z.B. die Konstruktion zum vorhergehenden Satz bzw. Übungsaufgabe. □

Satz 104

Die Familie CSL der kontextsensitiven (bzw. Chomsky-1) Sprachen ist abgeschlossen unter den folgenden Operationen:

$$\cap\,,\ \cup\,,\ \cdot\,,\ ^*\,,\ ^-$$

Beweis:

Der Beweis für die ersten vier Operationen ergibt sich unmittelbar aus den Eigenschaften linear beschränkter Automaten, der Abschluss unter Komplement wird später gezeigt („Induktives Zählen“). . ☐

Satz 104

Die Familie CSL der kontextsensitiven (bzw. Chomsky-1) Sprachen ist abgeschlossen unter den folgenden Operationen:

$$\cap\,,\ \cup\,,\ \cdot\,,\ ^*\,,\ ^-$$

Beweis:

Der Beweis für die ersten vier Operationen ergibt sich unmittelbar aus den Eigenschaften linear beschränkter Automaten, der Abschluss unter Komplement wird später gezeigt („Induktives Zählen“). . □

Satz 105

Folgende Probleme sind für die Familie der Chomsky-1-Sprachen nicht entscheidbar:

1. *Leerheit*
2. *Äquivalenz*
3. *Durchschnitt*

Beweis:

ohne Beweis. □

Satz 105

Folgende Probleme sind für die Familie der Chomsky-1-Sprachen nicht entscheidbar:

1. *Leerheit*
2. *Äquivalenz*
3. *Durchschnitt*

Beweis:

ohne Beweis. □

Satz 105

Folgende Probleme sind für die Familie der Chomsky-1-Sprachen nicht entscheidbar:

1. *Leerheit*
2. *Äquivalenz*
3. *Durchschnitt*

Beweis:

ohne Beweis. □

Satz 105

Folgende Probleme sind für die Familie der Chomsky-1-Sprachen nicht entscheidbar:

1. *Leerheit*
2. *Äquivalenz*
3. *Durchschnitt*

Beweis:

ohne Beweis. □

Satz 105

Folgende Probleme sind für die Familie der Chomsky-1-Sprachen nicht entscheidbar:

1. *Leerheit*
2. *Äquivalenz*
3. *Durchschnitt*

Beweis:

ohne Beweis. $\square$

5.3 Chomsky-0-Sprachen

Satz 106

Zu jeder (nichtdeterministischen) TM N gibt es eine deterministische TM (DTM) M mit

$$L(N) = L(M)\,.$$

Beweis:

Die DTM erzeugt in BFS-Manier, für $k = 0, 1, \ldots$, alle Konfigurationen, die die TM N in k Schritten erreichen kann. Sie hält gdw sich dabei eine akzeptierende Konfiguration ergibt. □

5.3 Chomsky-0-Sprachen

Satz 106

Zu jeder (nichtdeterministischen) TM N gibt es eine deterministische TM (DTM) M mit

$$L(N) = L(M)\,.$$

Beweis:

Die DTM erzeugt in BFS-Manier, für $k = 0, 1, \ldots$, alle Konfigurationen, die die TM N in k Schritten erreichen kann. Sie hält gdw sich dabei eine akzeptierende Konfiguration ergibt. □

Satz 107

Die von (nichtdeterministischen oder deterministischen) Turingmaschinen akzeptierten Sprachen sind genau die Chomsky-0-Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

Satz 107

Die von (nichtdeterministischen oder deterministischen) Turingmaschinen akzeptierten Sprachen sind genau die Chomsky-0-Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

Satz 107

Die von (nichtdeterministischen oder deterministischen) Turingmaschinen akzeptierten Sprachen sind genau die Chomsky-0-Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.

„$\Longrightarrow$“: Die Grammatik erzeugt zunächst alle akzeptierenden Konfigurationen (der Form $\alpha q_f \beta$ mit $q_f \in F$). Sie enthält weiterhin Regeln, die es gestatten, aus jeder Satzform, die eine Konfiguration darstellt, alle möglichen unmittelbaren Vorgängerkonfigurationen abzuleiten. Die zur Anfangskonfiguration (ϵ, q_0, w) gehörige Satzform ist damit ableitbar gdw die TM das Wort w akzeptiert.

Die Produktionen ergeben sich kanonisch aus der Übergangsrelation der TM. Sie sind i.a. nicht mehr längenmonoton!

Satz 107

Die von (nichtdeterministischen oder deterministischen) Turingmaschinen akzeptierten Sprachen sind genau die Chomsky-0-Sprachen.

Beweis:

Wir beschreiben nur die Beweisidee.
„$\Longleftarrow$“: Wir simulieren mit der TM alle Ableitungen der Chomsky-0-Grammatik in BFS-Manier und akzeptieren, falls eine solche Ableitung das Eingabewort x ergibt.
Man beachte, dass die konstruierte TM nicht unbedingt immer hält!
Eine andere Idee ist, wie im LBA-Fall die Ableitungen rückwärts zu simulieren. $\square$