Vorlesung Theoretische Informatik

Automaten und Formale Sprachen

Hochschule Reutlingen
Fakultät für Informatik
Masterstudiengang Wirtschaftsinformatik
überarbeitet von F. Laux (Stand: 09.06.2010)

Sommersemester 2010

Foliensatz - Inhaltsverzeichnis

Einleitung

Formale Sprachen sind Grundlage der Automatentheorie und der Theorie der Berechenbarkeit und Komplexität!

- Grammatiken erzeugen Formale Sprachen.
- Automaten und Turing-Maschinen erkennen Formale Sprachen.
- Entscheidungsprobleme sind Formale Sprachen.

Begriffe und Definitionen

Alphabet, Wort, Zeichen

Ein Alphabet ist eine endliche, nicht leere Menge Σ. Eine Zeichenkette $w = x_1 x_2 \dots x_n$ aus Zeichen $x_k \in \Sigma$ heißt Wort (über dem Alphabet Σ).

Wortlänge, Konkatenation

Die Länge $|w|$ eines Wortes w ist die Anzahl Zeichen, aus denen w besteht. Für Wörter $w_1 = x_1 x_2 \dots x_m$, $w_2 = y_1 y_2 \dots y_n$ heißt das Wort $w_1 w_2 = x_1 x_2 \dots x_m y_1 y_2 \dots y_n$ Konkatenation der Wörter w_1, w_2.

DasleereWort

Das Wort ϵ der Länge 0 heißt leeres Wort.

Begriffe und Definitionen

Syntax für weitere Operationen

Für ein Wort w ist w^R das Wort w umgedreht, z.B. $(abc)^R = cba$. Mit w^n bezeichnen wir die n-fache Konkatenation von w, z.B. $a^3 = aaa$. Dabei ist $w^0 = \epsilon$.

Konkatenation von Mengen

Die Konkatenation von Mengen A, B ist $AB = \{ab | a \in A, b \in B\}$.

Konkatenationsabschluss

= Transitiver Abschluss bezügl. Konkatenation

Der Konkatenationsabschluss Σ^* ist die Menge aller Wörter über Σ, einschließlich dem leeren Wort.

Begriffe und Definitionen

Definition Formale Sprache

Eine Formale Sprache L über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Begriffe und Definitionen

Eine lexikografische Ordnung einer Menge A ist eine Folge (w_n) von Wörtern, für die gilt:

- Die Wortlänge $|w_n|$ steigt monoton.
- Innerhalb von Abschnitten mit gleichen Wortlängen sind die Glieder der Folge alphabetisch (d.h. Σ hat Ordnungsrelation) geordnet.

Begriffe und Definitionen

Komplement

Für eine Menge A bezeichnen wir mit $\overline{A}$ das Komplement von A bezüglich Σ^*.
$A \subseteq \Sigma^*, \overline{A} = \Sigma^* \setminus A$

Grammatiken

- Grammatiken beschreiben formale Sprachen mit endlich vielen Regeln.
- Durch die Anwendung der Regeln können Wörter gebildet werden, die in ihrer Gesamtheit die Sprache bilden.
- Formale Sprachen werden von einer Grammatik vollständig beschrieben.
- Nachfolgend werden verschiedene Klassen von Grammatiken und Sprachen betrachtet.

Grammatiken

Eine Grammatik besteht aus

- Variablen; eine der Variablen ist das Startsymbol S.
- Alphabet (die Elemente werden auch Terminale genannt).
- Ersetzungsregeln (oder Produktionsregeln).

Form der Regeln

$$\textit{linke Seite} \longrightarrow \textit{rechte Seite}$$

Grammatiken

Formal ist eine Grammatik $G = (V, \Sigma, S, P)$ definiert durch

- die endliche Menge der Variablen V
- das Alphabet Σ mit $V \cap \Sigma = 0$
- das Startsymbol $S \in V$
- die endliche Menge von Regeln (Produktionen) P der Form $u \longrightarrow v$, wobei u mindenstens eine Variable enthält: $u \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$, $v \in (V \cup \Sigma)^*$.

Grammatiken

Sei $x, y \in (V \cup \Sigma)^*$.

Ableitung in einem Schritt

$x \Rightarrow y$ gilt genau dann, wenn es eine Regel $u \longrightarrow v$ gibt, so dass $x = pus$,$y = pvs$ für $p, s \in (V \cup \Sigma)^*$.

Ableitung in beliebig vielen Schritten

Wir schreiben $x \Rightarrow^* y$, wenn sich aus x in beliebig vielen (auch 0) Schritten y ableiten lässt.

Die aus G resultierende Sprache $L(G)$ ist definiert durch:

$L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$

Chomsky-Hierarchie

Grammatiken können nach Noam Chomsky in Typen, nämlich Typ 0-3 eingeteilt werden:

Typ 0

Jeder Typ ist zunächst automatisch vom Typ 0. Es gibt keine Einschränkungen bei den Regeln.

Typ 1

Eine Grammatik ist vom Typ 1 oder kontextsensitiv, falls für alle Regeln $u \longrightarrow v$ in P gilt $|u| \leq |v|$.

Chomsky-Hierarchie

Typ 2

Eine Typ 1-Grammatik ist vom Typ 2 oder kontextfrei, falls für alle Regeln $u \longrightarrow v$ gilt, dass u eine einzelne Variable ist.

Typ 3

Eine Typ 2-Grammatik ist vom Typ 3 oder regulär, falls zusätzlich gilt: $v \in \Sigma \cup \Sigma V$,d.h. die rechten Seiten von Regeln sind Terminale oder Terminale gefolgt von einer Variablen.

Chomsky-Hierarchie

Sprache vom Typ X

Eine Sprache $L \subseteq \Sigma^*$ heißt vom Typ 0 (Typ 1, Typ 2, Typ 3), falls es eine Typ 0 (Typ 1, Typ 2, Typ 3)-Grammatik G gibt mit $L(G) = L$.

Chomsky-Hierarchie

Die Definition der Chomsky-Hierarchie verhindert bei Grammatiken vom Typ 1-3 das Ableiten des leeren Wortes ($|u| \leq |v|$). Eine zusätzliche Regelung soll das beheben! Die Sprache soll dabei nicht verändert werden, bis auf die Tatsache, dass ϵ dazu gehört.

ϵ-Sonderregelung

Das leere Wort ϵ darf nur aus dem Startsymbol S abgeleitet werden. S darf auf keiner rechten Seite einer Regel vorkommen.

Chomsky-Hierarchie

Umwandlung von Typ 1 und 2 Grammatiken

1. Falls $S \longrightarrow \epsilon \in P$, dann ersetze alle S auf der rechten Seite durch ϵ und und füge die Ergebnisse als neue Regeln ein.
2. Neues Startsymbol S' und die Regel $S' \longrightarrow S$ einführen.
3. Umbenennen von $S \longrightarrow \epsilon$ in $S' \longrightarrow \epsilon$ (falls vorhanden) oder neu hinzufügen.

Beispiel:

- $s \longrightarrow \varepsilon \mid 0s1 \mid asb$
- $s' \longrightarrow \varepsilon \mid s$
  $s \longrightarrow 0s1 \mid asb \mid 01 \mid ab$

Chomsky-Hierarchie

Umwandlung von Typ 3 Grammatiken ($S \rightarrow S'$ ist nicht regulär!)

1. Falls $S \rightarrow \epsilon \in P$, dann ersetze alle S auf der rechten Seite durch ϵ und und füge die Ergebnisse als neue Regeln ein.
2. Jedes S auf der rechten Seite einer Regel wird durch S' ersetzt.
3. Kopiere alle Regeln $S \rightarrow \ldots$ und ersetze auf der linken Seite S durch S'.
4. Als letztes fügt man $S \rightarrow \epsilon$ ein (falls $\epsilon \in L(G)$ gelten soll) bzw. entfernt die Regel $S' \rightarrow \epsilon$ (falls ϵ bereits enthalten war).

$\rightarrow$ Beispiel

Chomsky-Hierarchie

1. In Typ 2 und 3 Grammatiken ist es auch erlaubt Regeln der Form $A \longrightarrow \epsilon$ (A ist keine Startvariable) zu haben.
2. Sind Regeln dieser Form enthalten, so muss der Schritt 1 in beiden Verfahren auch für diese Variablen angewandt werden.
3. Ggf. muss der Schritt mehrfach angewandt werden (z.B. bei $A \longrightarrow B$, $B \longrightarrow \epsilon$).

$\longrightarrow$ Beispiel

Chomsky-Hierarchie

Sei $\mathbb{L}_k$ die Menge aller Typ-k-Sprachen, dann gilt:

$$\mathbb{L}_3 \subset \mathbb{L}_2 \subset \mathbb{L}_1 \subset \mathbb{L}_0$$

regulär $\subset$ kontextfrei $\subset$ k.sensitiv $\subset$ beliebig

Für die praktische Umsetzung in der Informatik (Syntaxanalyse, Compilerbau) sind vor allem die regulären und kontextfreien Sprachen von Interesse.

Reguläre Sprachen

- Reguläre Sprachen (Typ 3) lassen sich durch
  1. reguläre Grammatiken,
  2. endliche Automaten sowie durch
  3. reguläre Ausdrücke beschreiben.
- Die endlichen Automaten gibt es in zwei Versionen:
  1. Deterministischer Endlicher Automat (DEA bzw. DFA (engl.)).
  2. Nicht-Deterministischer Endlicher Automat (NEA bzw. NFA (engl.)).

Reguläre Sprachen

Aufbau eines Endlichen Automaten läßt sich durch einen gerichteten Graphen beschreiben.

- Die Kanten sind mit einem Zeichen aus Σ beschriftet.
- Jeder Knoten des Graphen entspricht einem Zustand des Automaten.
- Der Startzustand ist durch einen eingehenden Pfeil gekennzeichnet.
- Endzustände sind durch zwei Kreise gekennzeichnet.

Reguläre Sprachen

Funktionsweise eines Endlichen Automaten.

- Die Verarbeitung beginnt im Startzustand.
- Nach jedem eingegebenen Zeichen wird zum Zustand gewechselt, der über die mit dem Zeichen markierte Kante erreicht werden kann.
- Eine Folge von Zeichen wird durch den Automaten akzeptiert, falls sich der Automat nach Eingabe des letzten Zeichens in einem Endzustand befindet.
- Der Automat speichert keine Informationen über die bisher gegangenen Wege.

Deterministischer Endlicher Automat

Ein DEA ist ein Automat, der nach einem gelesenen Zeichen in genau einen Folgezustand wechseln kann.

- Ein DEA M besitzt genau einen Startzustand und mindestens einen Endzustand.
- Die von M akzeptierte Sprache L(M) besteht aus allen Eingaben $w \in \Sigma^*$, so dass M nach dem Lesen von w einen Endzustand erreicht.

Reguläre Sprachen

Formal lässt sich ein DFA M=(Z,Σ,δ,z_0,E) definieren durch

- die Menge der Zustände Z
- das Eingabealphabet Σ
- die Überführungsfunktion $\delta : Z \times \Sigma \rightarrow Z$. Dabei bedeutet $\delta(z, a) = z'$, dass M im Zustand z für die Eingabe a in den Zustand z' wechselt.
- den Startzustand $z_0 \in Z$
- die Menge der Endzustände $E \subseteq Z$

Beispiel: Lichtschalter M = (Z, Σ, δ, aus, E)
wobei: Z = {aus, ein} = E, Σ = {drücken}, z_0 = aus

δ: $Z \times \Sigma \rightarrow Z$

(aus,drücken) $\mapsto$ ein

(ein,drücken) $\mapsto$ aus

Reguläre Sprachen

Mit Hilfe der durch

$$\hat{\delta}(z, \epsilon) = z$$

$$\hat{\delta}(z, a_1 a_2 \dots a_k) = \delta(\hat{\delta}(z, a_1 a_2 \dots a_{k-1}), a_k)$$

für $a_1 a_2 \dots a_k \in \Sigma^*$ definierten erweiterten Überführungsfunktion $\hat{\delta} : Z \times \Sigma^* \longrightarrow Z$ lässt sich die von M akzeptierte Sprache definieren als

$$L(M) = \{w \in \Sigma^* | \hat{\delta}(z_0, w) \in E\}.$$

Das heißt, $L(M)$ besteht aus allen Wörtern w, so dass M, gestartet im Zustand z_0 mit Eingabe w, einen Endzustand erreicht.

Reguläre Sprachen

Aus einem DFA M lässt sich sehr einfach eine reguläre Grammatik G mit $L(G) = L(M)$ erzeugen.

- Die Zustände entsprechen den Variablen:
  - Für jeden Zustand z führen wir eine Variable Z ein.
  - Das Startsymbol der Grammatik ist Z_0.
- Die Übergänge entsprechen den Regeln:
  - Für jeden Übergang $\delta(z, a) = z'$ führen wir die Regel $Z \rightarrow aZ'$ ein.
  - Für jeden Endzustand führen wir die Regel $Z \rightarrow \epsilon$ ein.

Ein NEA (Nichtdeterministischer endlicher Automat; Nondeterministic Finite Automaton; NFA (engl.)) ist eine Verallgemeinerung des DEA.

NEA vs. DEA

Ein NEA kann nach jedem gelesenen Zeichen in beliebig viele Folgezustände wechseln. Die Anzahl der Folgezustände kann auch 0 sein $\Rightarrow$ Sackgasse! Ein NEA besitzt einen oder mehrere Startzustände.

$\Rightarrow$ Beispiel.

Analog zu einem DEA akzeptiert ein NEA die folgenden Sprachen:

Sprache eines NEA

Die von einem NEA M akzeptierte Sprache L(M) besteht aus allen Eingaben $w \in \Sigma^*$, so dass M, gestartet in einem Startzustand, nach dem Lesen von w einen Endzustand erreichen kann.

Die Überführungsfunktion des DEA kann wie folgt angepasst werden:

Überführungsfunktion δ für einen NEA

Für einen NEA M und $z \in Z, a \in \Sigma$ ist $\delta(z, a)$ eine Menge von Zuständen, die auch leer sein kann. Für jedes $a \in \Sigma$ wechselt M in einen der Zustände aus $\delta(z, a)$.

- Für $\delta(z, a) = \{\}$ gibt es damit keinen Zustand, in den M wechseln kann, so dass M seine Arbeit einstellt.
- Alternativ: Betrachtung als Übergang in einen Fehlerzustand, der nicht mehr verlassen werden kann.
- DEA ist Spezialfall eines NEA mit einem Startzustand und $|\delta(z, a)| = 1$ für alle $z \in Z, a \in \Sigma$.

Reguläre Sprachen

Aus einer regulären Grammatik G kann ein NEA M mit $L(M) = L(G)$ erzeugt werden:

- Die Variablen entsprechen den Zuständen,
- die Regeln den Übergängen des NEA.
- Falls $A \longrightarrow \epsilon$ eine Regel der Grammatik ist, so ist der A entsprechende Zustand ein Endzustand.
- Regeln der Form $A \longrightarrow a$ werden durch $A \longrightarrow aA', A' \longrightarrow \epsilon$ ersetzt.

$\Rightarrow$ Beispiel.

Reguläre Sprachen

- Ein NEA ist ein nützliches Instrument, mit dem der Umgang leichter ist als mit einem DEA.
- Der Nichtdeterminismus ist jedoch nicht geeignet für eine praktische Anwendung.

$\Rightarrow$ Abhilfe schafft die Umwandlung des NEA in einen DEA!

Reguläre Sprachen

- Ein NEA ist ein nützliches Instrument, mit dem der Umgang leichter ist als mit einem DEA.
- Der Nichtdeterminismus ist jedoch nicht geeignet für eine praktische Anwendung.

$\Rightarrow$ Abhilfe schafft die Umwandlung des NEA in einen DEA!

Umwandlung NEA $\rightarrow$ DEA

Prinzip

Zusammenfassen von mehreren Zuständen des NEA $\{z_{i_1}, z_{i_2}, \ldots\}$ zu einem Zustand des DEA $z = \{z_{i_1}, z_{i_2}, \ldots\}$

- Die Startzuständen des NEA, werden zum Startzustand des DEA zusammengefasst.
- Für jeden neuen Zustand des DEA und jedes Zeichen fassen wir die durch das Zeichen erreichbaren Zustände zu einem neuen Zustand des DEA zusammen.
- Enthält ein neuer Zustand des DEA einen Endzustand des NEA, dann ist auch er ein Endzustand des DEA.
- Fehlenden Übergänge führen in einen Fehlerzustand.

Formal ergibt sich die Überführungsfunktion δ' des DEA aus der Überführungsfunktion δ des NEA

$$\delta'(z,a) = \begin{cases} \bigcup_{s \in z} \delta(s,a) \text{ für } \bigcup_{s \in z} \delta(s,a) \neq \{\} \\ \{F\} \text{ sonst} \end{cases}$$

, wobei F der Fehlerzustand ist und $a \in \Sigma$.

Minimalautomaten

Bei der Umwandlung kann der DEA durch Teilmengenkonstruktion wesentlich mehr Zustände besitzen als der NEA. Oft sind dabei überflüssige Zustände enthalten, die sich durch eine Minimierung beseitigen lassen. Das Resultat ist der **Minimalautomat**.

Bei der Konstruktion werden Paare von nicht äquivalenten Zuständen markiert und die verbleibenden äquivalenten Zustände werden zusammengefasst.

Äquivalenz von Zuständen:
Zwei Zustände z, z' sind nicht äquivalent, wenn

- entweder z oder z' ein Endzustand ist oder
- wenn für ein Zeichen $a \in \Sigma$ die Folgezustände $\delta(z, a)$, $\delta(z', a)$ nicht äquivalent sind.

$\Rightarrow$ Beispiel für zwei nicht äquivalente Zustände.

Reguläre Sprachen

Zur Bestimmung sämtlicher Paare mit nicht-äquivalenten Zuständen stellen wir eine **Tabelle** von ungeordneten Paaren von Zuständen auf:

- Es werden alle Paare betrachtet und diejenigen markiert, die aus nicht äqivalenten Zuständen bestehen.
- Das wird so lange gemacht bis sich an der Tabelle nicht mehr ändert.
- Die nicht markierten Paare sind äquivalent und können zusammengefasst werden.
- Zustände, die nicht vom Startzustand aus erreichbar sind werden vorher entfernt!

$\Rightarrow$ Beispiel.

Reguläre Sprachen

Äquivalente Zustände können auch mit der erweiterten Übergangsfunktion $\hat{\delta}$ beschrieben werden. Die Zustände z und z' sind äquivalent, wenn für alle $w \in \Sigma^*$ gilt:

$$\hat{\delta}(z, w) \in E \iff \hat{\delta}(z', w) \in E$$

D.h., für kein $w \in \Sigma^*$ macht es einen Unterschied, ob der DEA in z oder z' startet.