Theoretische Informatik für Wirtschaftsinformatik und Lehramt

Reguläre Sprachen

Priv.-Doz. Dr. Stefan Milius
stefan.milius@fau.de

Theoretische Informatik
Friedrich-Alexander Universität Erlangen-Nürnberg

SS 2016

Gliederung

Worum geht es in diesem Abschnitt? (I)

- **Erste Formalismen** zur endlichen algorithmischen Beschreibung von formalen Sprachen
  (= Lösung der entsprechenden Entscheidungsprobleme):

  Automaten und Grammatiken

- **Praktische Anwendungen:**
  - Compilerbau: lexikalische Analyse, Parsing, Textprocessing
  - Automatische Systemanalyse: z.B. formale Verifikation von nicht-trivialen Eigenschaften (Modelchecking)

- **Chomsky-Hierarchie:** Hierarchie von Klassen von formalen Sprachen und Grammatiken (und Maschinenmodellen)

  (setzt die verschiedenen Sprachklassen in Beziehung zueinander)

Noam Chomsky

Amerikanischer Linguist, Philosoph, Logiker, Historiker und politischer Aktivist

- Vater der modernen Linguistik
- Analytische Philosophie
- Über 100 Bücher

Noam Chomsky (1928–)
Quelle: Wikipedia

Bekannte Entdeckungen:

- Chomsky-Hierarchie
- Chomsky-Schützenberger Satz
- Einfluss auf:

künstliche Intelligenz, Kognitionswissenschaft, Mathematik, Logik, Informatik, Programmiersprachentheorie, Psychologie und Politikwissenschaft.

Worum geht es in diesem Abschnitt? (II)

Jetzt als Erstes: Reguläre Sprachen

- besonders einfach
- z.B. Suche nach regulären Ausdrücken in Texten
- Mechanismen zur Erzeugung/Beschreibung regulärer Sprachen:
  - reguläre Grammatiken
  - reguläre Ausdrücke
  - deterministische und nichtdeterministische endliche Automaten

Lernziele

- wissen, wie eine reguläre Sprache erzeugt und akzeptiert werden kann
- die Chomsky-Hierarchie kennen und Eigenschaften der ihr zugeordneten Grammatik-Typen benennen können

Grammatiken

Zur Erinnerung: Formale Sprachen

Definition

Sei Σ ein Alphabet. (d.h. eine nicht leere, endliche Menge)
Eine (formale) Sprache L über Σ ist eine beliebige Teilmenge von Σ^* (= Menge aller Wörter über Σ):

$$L \subseteq \Sigma^*.$$

Die leere Teilmenge $\emptyset$ heißt leere Sprache.

Zu jeder Sprache L über Σ gehört das Entscheidungsproblem:

- **Eingabe:** $w \in \Sigma^*$
- **Aufgabe:** entscheide, ob $w \in L$ gilt.

Formale Sprachen

Beispiel

$\Sigma = \{(,), +, -, *, /, a\}$

Sprache der korrekt geklammerten arithmetischen Ausdrücke:

$$EXPR \subseteq \Sigma^*$$

$$\begin{aligned} (a - a) * a + a/(a + a) - a &\in EXPR \\ (((a))) &\in EXPR \\ ((a+) - a(&\notin EXPR \end{aligned}$$

(a Platzhalter für beliebige Konstanten oder Variablen)

Grammatiken (I)

- Sprachen enthalten im Allgemeinen unendlich viele Wörter
- algorithmischer Umgang mit Sprachen erfordert endliche Beschreibung
- dazu dienen Grammatiken und Automaten

Beispiel für Grammatik

$$
\begin{array}{lcl}
\langle \text{Satz} \rangle & \to & \langle \text{Subjekt} \rangle \ \langle \text{Prädikat} \rangle \ \langle \text{Objekt} \rangle \\
\langle \text{Subjekt} \rangle & \to & \langle \text{Artikel} \rangle \ \langle \text{Attribut} \rangle \ \langle \text{Substantiv} \rangle \\
\langle \text{Artikel} \rangle & \to & \varepsilon \\
\langle \text{Artikel} \rangle & \to & \text{der} \\
\langle \text{Artikel} \rangle & \to & \text{die} \\
\langle \text{Artikel} \rangle & \to & \text{das} \\
\langle \text{Attribut} \rangle & \to & \varepsilon \\
\langle \text{Attribut} \rangle & \to & \langle \text{Adjektiv} \rangle \\
\langle \text{Attribut} \rangle & \to & \langle \text{Adjektiv} \rangle \ \langle \text{Attribut} \rangle
\end{array}
$$

Grammatiken (II)

Beispiel für Grammatik (Fortsetzung)

$$
\begin{array}{lcl}
\langle \text{Adjektiv} \rangle & \to & \text{kleine} \\
\langle \text{Adjektiv} \rangle & \to & \text{bissige} \\
\langle \text{Adjektiv} \rangle & \to & \text{große} \\
\langle \text{Substantiv} \rangle & \to & \text{Hund} \\
\langle \text{Substantiv} \rangle & \to & \text{Katze} \\
\langle \text{Prädikat} \rangle & \to & \text{jagt} \\
\langle \text{Objekt} \rangle & \to & \langle \text{Artikel} \rangle \ \langle \text{Attribut} \rangle \ \langle \text{Substantiv} \rangle
\end{array}
$$

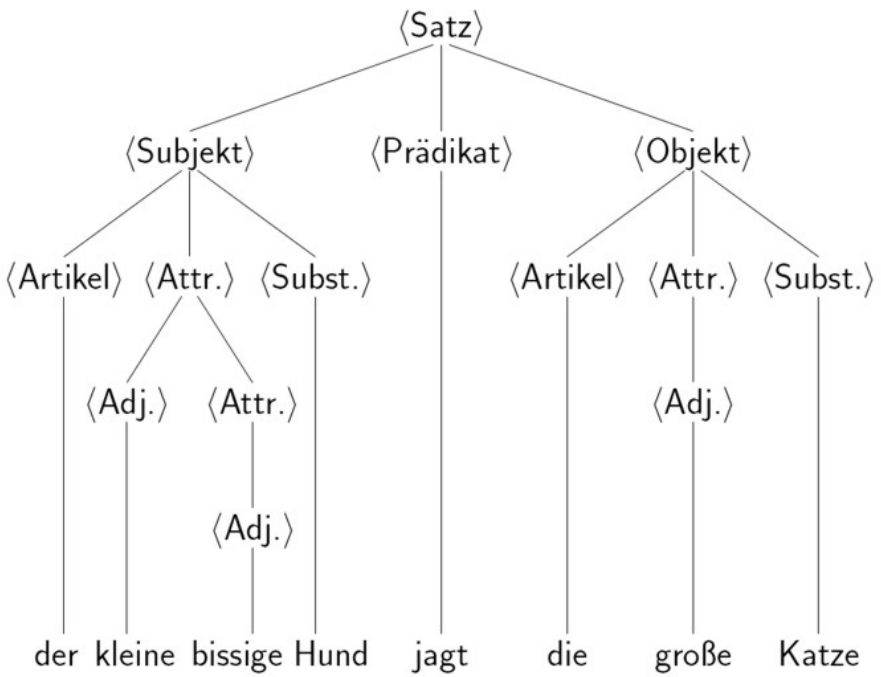
- in spitzen Klammern $\langle \cdots \rangle$: Nicht-Terminale (oder Variable)
- ohne spitze Klammern: Terminale

Frage: Gehört der Satz

„der kleine bissige Hund jagt die große Katze“

zu der Sprache, die von der Grammatik erzeugt wird?

Syntaxbaum



- Elternknoten mit linker Seite einer Regel beschriftet
- Kinder sind Objekte auf der rechten Seite der Regeln
- mit dieser Grammatik ist unendliche Sprache darstellbar, z.B.

„der Hund jagt die $\underbrace{\text{kleine kleine} \cdots \text{kleine}}_{\text{beliebige endliche Anzahl}}$ Katze“

Grammatiken (III)

Grammatiken besitzen Regeln der Form

$$\langle \text{linke Seite} \rangle \rightarrow \langle \text{rechte Seite} \rangle$$

Links und rechts zwei Typen von Symbolen:

- Nicht-Terminale (oder Variablen) $\cdots\cdots$ aus denen noch weitere Wortbestandteile abgeleitet werden sollen
- Terminale $\cdots\cdots$ die eigentlichen Symbole

Vorheriges Beispiel:
Auf der linken Seite befindet sich immer genau ein Nicht-Terminal (sog. kontextfreie Grammatik).

Es gibt aber allgemeinere Grammatiken.

Definition einer Grammatik

Definition

Eine **Grammatik** ist ein 4-Tupel $G = (V, \Sigma, P, S)$ mit

- V endliche Menge der **Nicht-Terminale** (oder **Variablen**)
- Σ Alphabet der **Terminale**
- P endliche Menge der **Regeln** (oder **Produktionen**):

$$P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$$

- $S \in V$ **Startvariable**.

Schreibweisen

- Variablen (Elemente aus V) mit Großbuchstaben:

$$A, B, C, \ldots$$

- Terminalsymbole (Elemente aus Σ) oft mit Kleinbuchstaben:

$$a, b, c, \ldots$$

- Wörter (Elemente aus Σ^*) mit Kleinbuchstaben um „w herum“:

$$\ldots, u, v, w, \ldots$$

- Regeln mit einem „Pfeil“:

$$u \to v \qquad \text{statt} \qquad (u, v) \in P.$$

(d.h. $\to \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ Relation)

Beispiel einer Grammatik

Beispiel

$G = (V, \Sigma, P, S)$ mit

$$V = \{S, B, C\}, \quad \Sigma = \{a, b, c\} \quad \text{und } P \text{ besteht aus:}$$

$S \to aSBC$	$aB \to ab$	$cC \to cc$
$S \to aBC$	$bB \to bb$	
$CB \to BC$	$bC \to bc$	

Frage:

Wie beschreiben Grammatiken Wörter formaler Sprachen?

Ableitungen (I)

Definition

Seien $u, v \in (V \cup \Sigma)^*$.
Definiere die Relation $u \Rightarrow_G v$ $(\Rightarrow_G \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma^*))$
(„v aus u mit G direkt ableitbar“) falls

$$u = xyz \quad \text{und} \quad v = xy'z \quad \text{mit } x, y \in (V \cup \Sigma)^* \text{ und } y \to y' \text{ Regel in } P.$$

$\Rightarrow_G^*$ sei die reflexive, transitive Hülle von $\Rightarrow_G$, d.h.:
$u \Rightarrow_G^* v$ falls es $w_0, w_1, w_2, \ldots, w_n$ $(n \in \mathbb{N})$ gibt mit

$$u = w_0 \Rightarrow_G w_1 \Rightarrow_G w_2 \Rightarrow_G \cdots \Rightarrow_G w_n = v$$

(„v aus u mit G ableitbar“) (in $n = 0, 1, 2, \ldots$ Schritten)
Die Folge $w_0, w_1, \ldots, w_n$ heißt Ableitung (von v aus u).

Wir schreiben meist: $\Rightarrow$ bzw. $\Rightarrow^*$ statt $\Rightarrow_G$ und $\Rightarrow_G^*$.

Ableitungen (II)

Beispiel

Für $G = (V, \Sigma, P, S)$ mit $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$ und P:

$S \to aSBC$	$aB \to ab$	$cC \to cc$
$S \to aBC$	$bB \to bb$	
$CB \to BC$	$bC \to bc$	

Es gilt: $S \Rightarrow^* aaabbbccc$, denn

$$\begin{aligned} \underline{S} &\Rightarrow a\underline{S}BC \Rightarrow aa\underline{S}BCBC \Rightarrow aaaB\underline{CB}CBC \\ &\Rightarrow aaaBBC\underline{CB}C \Rightarrow aaaBB\underline{CB}CC \Rightarrow aa\underline{aB}BBCCC \\ &\Rightarrow aaa\underline{bB}BCCC \Rightarrow aaab\underline{bB}CCC \Rightarrow aaabb\underline{bC}CC \\ &\Rightarrow aaabbb\underline{cC}C \Rightarrow aaabbbc\underline{cC} \Rightarrow aaabbbccc \end{aligned}$$

Es gilt auch: $aSBC \Rightarrow^* aaabbBCCC$.

Erzeugte Sprache einer Grammatik (I)

Definition

Sei $G = (V, \Sigma, P, S)$ eine Grammatik.

G erzeugt ein Wort $w \in \Sigma^*$, falls $S \Rightarrow^* w$ gilt.

Die von G erzeugte Sprache ist

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

(*genau* alle von G erzeugten Wörter über Σ)

Erzeugte Sprache einer Grammatik (II)

Beispiel

$G = (V, \Sigma, P, S)$ mit $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$ und P:

$S \to aSBC$	$aB \to ab$	$cC \to cc$
$S \to aBC$	$bB \to bb$	
$CB \to BC$	$bC \to bc$	

(Ohne Beweis) G erzeugt die Sprache:

$$L(G) = \{a^n b^n c^n \mid n \geq 1\}.$$

Ableiten und Nichtdeterminismus (I)

Bemerkung:

- Ableiten ist ein nichtdeterministischer Prozess.
- Für ein Wort $u \in (V \cup \Sigma)^*$ kann es entweder
  - gar kein,
  - ein oder
  - mehrere

  v geben mit $u \Rightarrow v$.
- Mit anderen Worten:

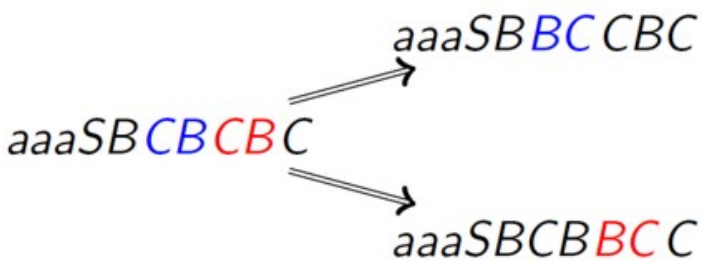
$$\Rightarrow \text{ ist } \textit{keine} \text{ Funktion.}$$

Dieser Nichtdeterminismus kann durch zwei verschiedene Effekte verursacht werden ...

Ableiten und Nichtdeterminismus (II)

- Zwei verschiedene Regeln sind anwendbar:
  In der Beispiel-Ableitung:

- Eine Regel ist an zwei verschiedenen Stellen anwendbar:
  In der Beispiel-Ableitung:



Weitere Bemerkungen zum Ableiten

- Es kann beliebig lange Pfade geben, die nie zu einem Wort aus Terminalsymbolen führen:

$$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaSBCBCBC \Rightarrow \cdots$$

- Pfade können in einer „Sackgasse “enden,
d.h. obwohl noch Variablen vorkommen, ist keine Regel mehr anwendbar:

$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabcBC \not\Rightarrow$$

Die Chomsky-Hierarchie (I)

- Allgemeine Grammatiken (und auch später Turing-Maschinen) erzeugen genau die sogenannten semi-entscheidbare Sprachen.
- Entscheidbare Sprachen nur dann, wenn die Erzeugungs-„Mächtigkeit“ der Grammatiken eingeschränkt wird.
- **Im Folgenden:** bestimmte eingeschränkte Typen von Grammatiken:

Die Chomsky-Hierarchie (II)

Definition (Typ 0-, Typ 1-, Typ 2-, Typ 3-Grammatiken)

Typ 0: Jede Grammatik ist vom Typ 0.
(keine Einschränkungen der Regelform)

Typ 1: Eine Grammatik ist vom Typ 1 oder kontextsensitiv, falls alle Regeln in P von einer der folgenden Formen sind:

$$S \to \varepsilon \qquad \text{oder} \qquad uAv \to uwv$$

mit $A \in V$, $u, v \in (\Sigma \cup V)^*$ und $w \in (\Sigma \cup V)^+$.

Typ 2: Eine Grammatik ist vom Typ 2 oder kontextfrei, falls alle Regeln in P in der folgenden Form sind:

$$A \to w \qquad \text{mit} \qquad A \in V, w \in (\Sigma \cup V)^*.$$

Die Chomsky-Hierarchie (III)

Definition (Typ 0-, Typ 1-, Typ 2-, Typ 3-Grammatiken)

Typ 3: Eine Grammatik ist vom Typ 3 oder regulär, falls alle Regeln in P in der folgenden Form sind:

$$A \to \varepsilon, \quad A \to a \quad \text{oder} \quad A \to aB \qquad \text{mit} \quad A, B \in V, a \in \Sigma.$$

(d.h. rechte Seiten entweder leer oder einzelne Terminalzeichen oder ein Terminalzeichen gefolgt von einer einzelnen Variablen.)

Woher kommen die Namen „kontextfrei „und „kontextsensitiv“?

- Bei kontextfreien Grammatiken: Regeln der Form $A \to w$:
  In Ableitungen $uAv \Rightarrow uwv$ wird A immer unabhängig vom Kontext ersetzt.
- Bei kontextsensitiven Grammatiken ist $uAv \to uwv$ möglich:
  A *kann nur* in Kontexten mit u und v durch w ersetzt werden.

Die Chomsky-Hierarchie (IV)

Schreibweise: Statt

$$A \to w_1, \quad A \to w_2, \quad \ldots, \quad A \to w_n \qquad \text{(d.h. gleiche linke Seite)}$$

schreibe: $A \to w_1 \mid w_2 \mid \cdots \mid w_n$.

Beispiele

- Typ 2- bzw. kontextfreie Grammatik

  $S \to aCb \mid C$
  $C \to c \mid Cc$

- Typ 3- bzw. reguläre Grammatik

  $S \to aS \mid aU$
  $U \to bB \mid b$

Die Chomsky-Hierarchie (V)

Definition (Typ 0-, Typ 1-, Typ 2-, Typ 3-Sprachen)

Eine Sprache $L \in \Sigma^*$ heißt

$$\text{vom Typ 0 (Typ 1, Typ 2, Typ 3)},$$

falls es eine Typ 0- (Typ 1-, Typ 2-, Typ 3-) Grammatik G gibt mit

$$L(G) = L.$$

Typ-2-Sprachen heißen auch kontextfrei.

Typ-3-Sprachen heißen auch regulär.

Die Chomsky-Hierarchie (VI)

Typ 1-Grammatiken sind vom Typ 0

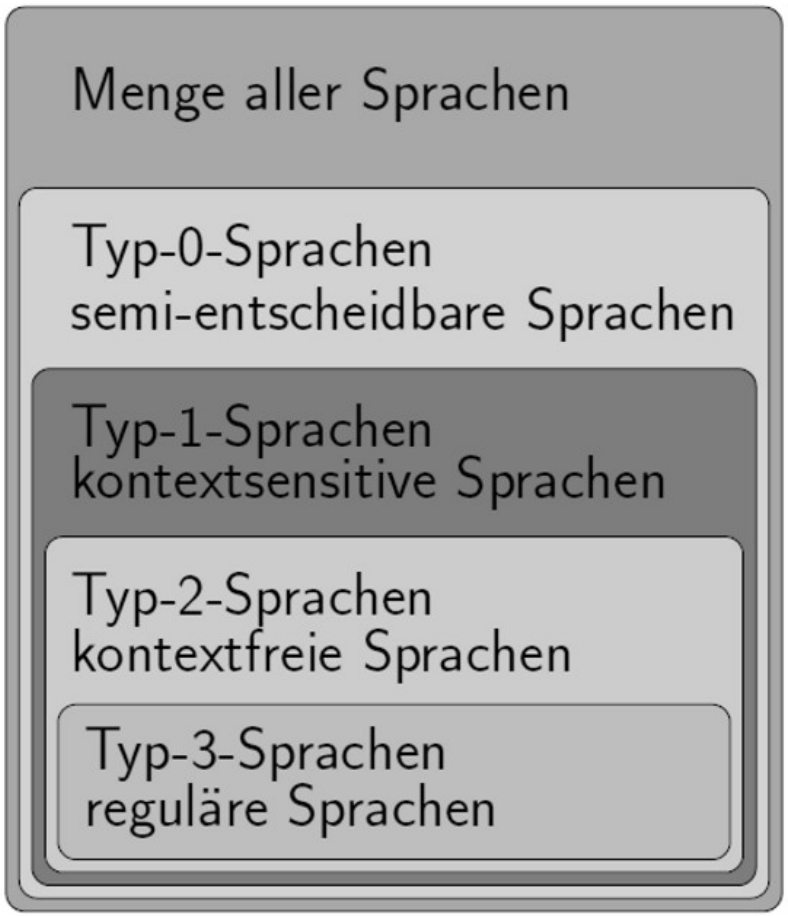
Typ 2-Grammatiken sind vom Typ 1
(können leicht umgewandelt werden)

Typ 3-Grammatiken sind vom Typ 2

$\Longrightarrow$ Sprachklassen ineinander enthalten

Außerdem: die Inklusionen sind echt.

(z.B. gibt es kontextfreie Sprachen, die nicht regulär sind)



Reguläre Sprachen

Reguläre Grammatiken (I)

Definition (Erinnerung)

Eine Grammatik $G = (V, \Sigma, P, S)$ heißt regulär, wenn jede Regel in P von einer der folgenden Formen ist:

$$A \to \varepsilon, \quad A \to a \quad \text{oder} \quad A \to aB \text{ (rechtslineare Regel)}$$

mit $A, B \in V$ und $a \in \Sigma$.

Beispiel einer regulären Grammatik

Grammatik G_{3a} mit $V = \{A, B, C\}, \Sigma = \{a, b\}, S = A$ und Regeln

$$\begin{aligned} A &\to aB \mid bA \mid \varepsilon, \\ B &\to aC \mid bB, \\ C &\to aA \mid bC \end{aligned}$$

Reguläre Grammatiken (II)

Beispiel einer Ableitung

Regeln:

$$A \to aB \mid bA \mid \varepsilon,$$
$$B \to aC \mid bB,$$
$$C \to aA \mid bC$$

Ableitung:

$$\begin{aligned} A &\Rightarrow bA \\ &\Rightarrow baB \\ &\Rightarrow baaC \\ &\Rightarrow baabC \\ &\Rightarrow baabaA \\ &\Rightarrow baaba \end{aligned}$$

Reguläre Sprachen (I)

Sei nun

$$L_{3a} = \{w \in \{a, b\}^* \mid |w|_a \text{ ist durch 3 teilbar}\}.$$

Behauptung: L_{3a} ist eine reguläre Sprache, denn es gilt

$$L_{3a} = L(G_{3a}).$$

Beweis: Für alle $w \in \{a, b\}^*$ gilt:

a) $A \Rightarrow^* w \iff |w|_a$ ist durch 3 teilbar.
b) $B \Rightarrow^* w \iff |w|_a + 1$ ist durch 3 teilbar.
c) $C \Rightarrow^* w \iff |w|_a + 2$ ist durch 3 teilbar.

Aus a) folgt dann $L_{3a} = L(G_{3a})$.

Reguläre Sprachen (II)

Beweis durch Induktion über den Aufbau von w:
(wir schreiben: $t_3(x)$ für „x ist durch 3 teilbar“):

1.) $w = \varepsilon$: es gilt $t_3(|w|_a)$ und

$$A \Rightarrow^* w, \quad \text{nicht } B \Rightarrow^* w \quad \text{und} \quad \text{nicht } C \Rightarrow^* w.$$

Daher gelten a), b) und c).

2) $w \neq \varepsilon$: dann ist $w = aw'$ oder $w = bw'$ $(w' \in \Sigma^*)$.
1. Fall: $w = aw'$. Dann gilt
(mit Anwendung der Induktionsvoraussetzung auf w'):

$$\begin{array}{lllllll}
(A \Rightarrow^* w) & \iff & (B \Rightarrow^* w') & \iff & t_3(|w'|_a + 1) & \iff & t_3(|w|_a), \\
(B \Rightarrow^* w) & \iff & (C \Rightarrow^* w') & \iff & t_3(|w'|_a + 2) & \iff & t_3(|w|_a + 1), \\
(C \Rightarrow^* w) & \iff & (A \Rightarrow^* w') & \iff & t_3(|w'|_a) & \iff & t_3(|w|_a + 2).
\end{array}$$

Reguläre Sprachen (III)

Beweis durch Induktion über den Aufbau von w (Fortsetzung):

2) $w \neq \varepsilon$ (Fortsetzung):

2. Fall: $w = bw'$. Dann gilt

(mit Anwendung der Induktionsvoraussetzung auf w'):

$$(A \Rightarrow^* w) \iff (A \Rightarrow^* w') \iff t_3(|w'|_a) \iff t_3(|w|_a),$$
$$(B \Rightarrow^* w) \iff (B \Rightarrow^* w') \iff t_3(|w'|_a + 1) \iff t_3(|w|_a + 1),$$
$$(C \Rightarrow^* w) \iff (C \Rightarrow^* w') \iff t_3(|w'|_a + 2) \iff t_3(|w|_a + 2).$$

In jedem Fall gelten also a), b) und c) für w. □

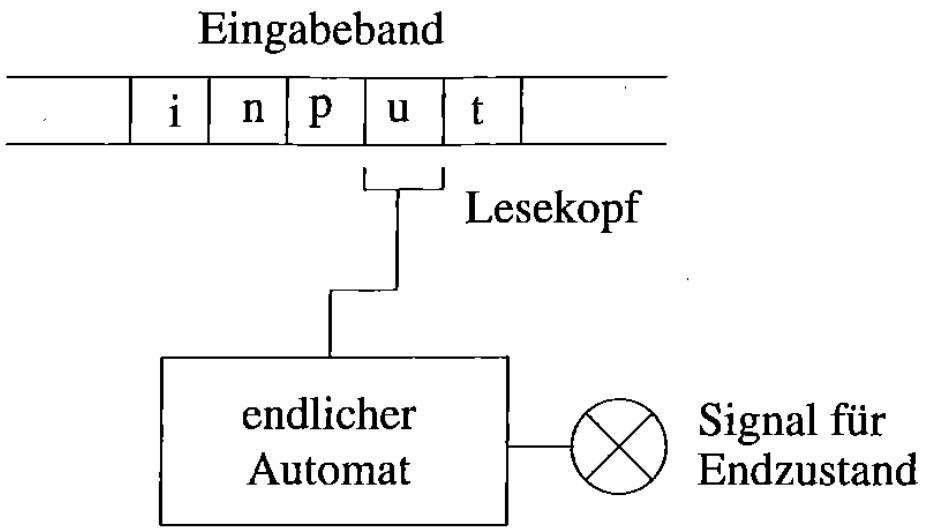
Deterministische endliche Automaten (DFA)

Deterministische endliche Automaten (I)

Bisher:

Grammatiken erzeugen Wörter formaler Sprachen aus „nichts“.

Jetzt: Verarbeitung von Wörtern durch abstrakte Maschinen:
endliche Automaten verarbeiten (akzeptieren) eingegebene Wörter



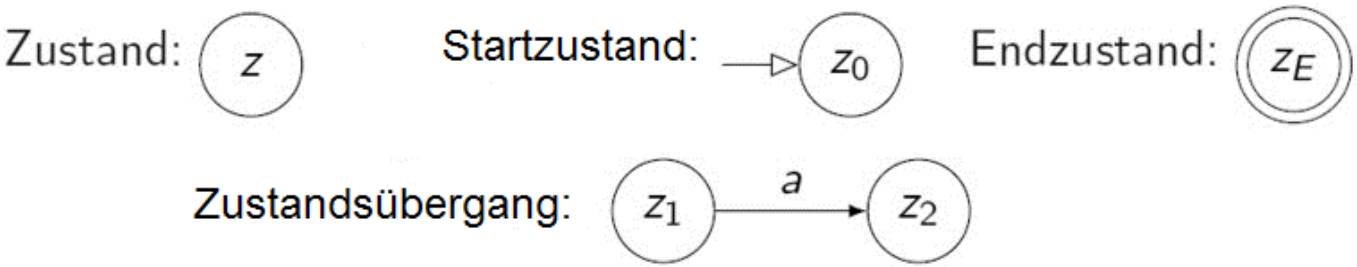
Deterministische endliche Automaten (II)

Arbeitsweise eines endlichen Automaten M:

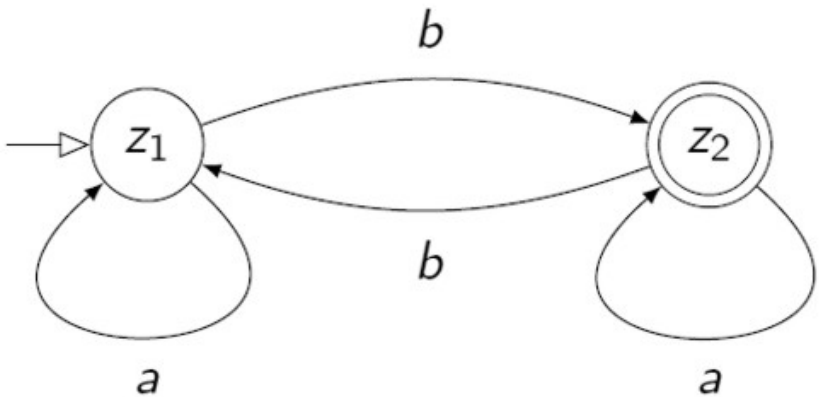
- liest die Eingabe von links nach rechts
- Symbole der Eingabe werden **nicht** verändert
- für jedes gelesene Symbol ändert M seinen internen Zustand
- Zustände zeigen an, ob (bis jetzt) gelesenes Wort akzeptiert/nicht akzeptiert wird

Deterministische endliche Automaten (III)

Graphische Notation:



Beispiel: $(\Sigma = \{a, b\})$



Deterministische endliche Automaten (IV)

Definition

Ein **deterministischer endlicher Automat (deterministic finite automaton, DFA)** ist ein 5-Tupel

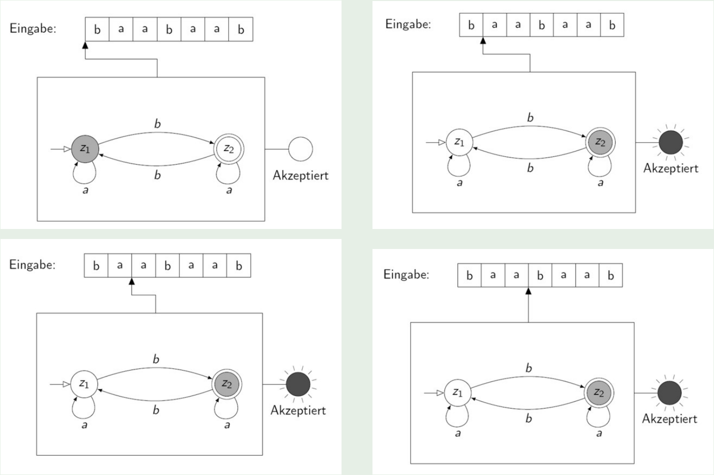
$$M = (Z, \Sigma, \delta, z_0, E) \qquad \text{gegeben durch}$$

- eine endliche Menge Z von **Zuständen**,
- ein **Eingabealphabet** Σ,
- eine totale **Überführungsfunktion** $\delta : Z \times \Sigma \to Z$,
- einen **Startzustand** $z_0 \in Z$,
- eine Menge $E \subseteq Z$ von **Finalzuständen** (auch **Endzustände** oder **akzeptierende Zustände**).

Warum deterministisch? Weil für jeden Zustand und jedes Eingabesymbol genau ein Folgezustand existiert.

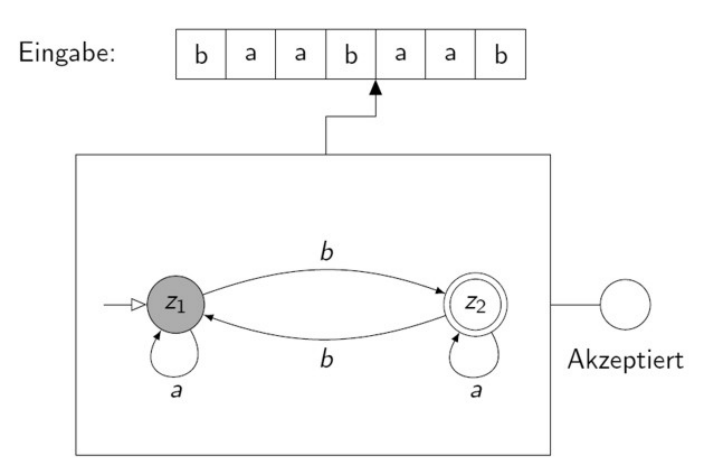
Ablauf eines DFA (I)

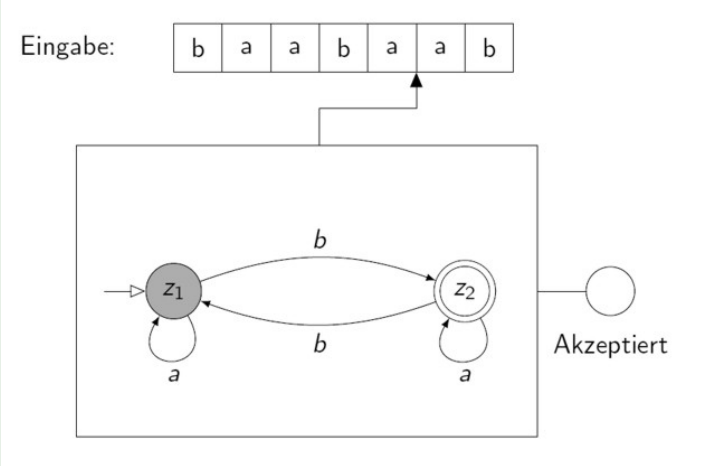
Beispiel

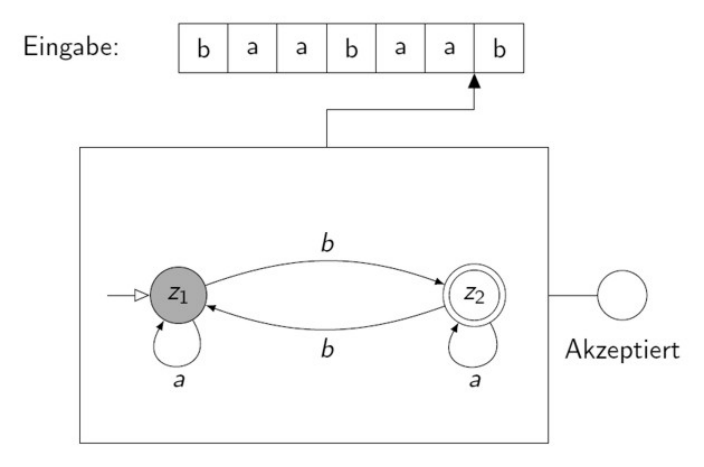


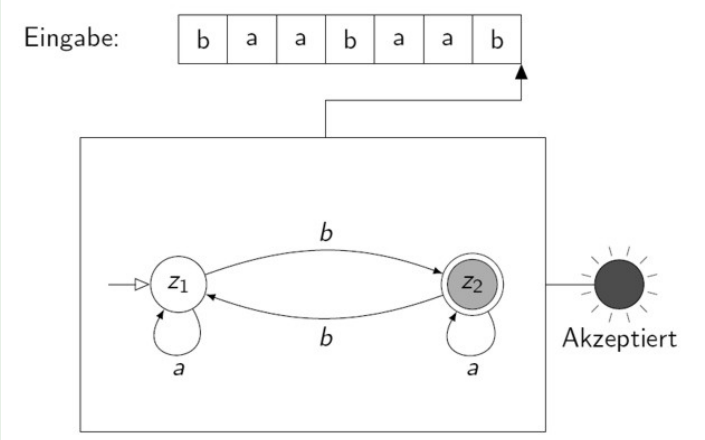
Ablauf eines DFA (II)

Beispiel (Fortsetzung)









Mehr-Schritt-Übergänge eines DFA (I)

Arbeitsweise und akzeptierte Wörter/Sprache eines DFA soll jetzt formalisiert werden.

Dazu:
Erweiterung der Übergangsfunktion $\delta : Z \times \Sigma \to Z$ auf Wörter:

$$\hat{\delta} : Z \times \Sigma^* \to Z,$$

Definition (Mehr-Schritt-Übergänge)

Zu einem gegebenen DFA $M = (Z, \Sigma, \delta, z_0, E)$ definieren wir eine Funktion $\hat{\delta} : Z \times \Sigma^* \to Z$ induktiv:

$$\begin{aligned} \hat{\delta}(z, \varepsilon) &= z, \\ \hat{\delta}(z, aw) &= \hat{\delta}(\delta(z, a), w) \qquad (z \in Z, a \in \Sigma, w \in \Sigma^*). \end{aligned}$$

Mehr-Schritt-Übergänge eines DFA (II)

$\hat{\delta}(z, w)$ ist der Zustand, den der Automat erreicht, nachdem er w „abgearbeitet“ hat.

Für $\hat{\delta}$ gilt offenbar:

a) $a \in \Sigma \Rightarrow \hat{\delta}(z, a) = \delta(z, a)$

b) $a_1, \ldots, a_n \in \Sigma \Rightarrow \hat{\delta}(z, a_1 \cdots a_n) = \delta(\cdots \delta(\delta(z, a_1), a_2), \ldots, a_n)$

c) $u, v \in \Sigma^* \Rightarrow \hat{\delta}(z, uv) = \hat{\delta}(\hat{\delta}(z, u), v)$

Akzeptierte Sprache eines DFA (I)

Definition

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA.

M **akzeptiert** ein Wort $w \in \Sigma^*$, falls $\hat{\delta}(z_0, w) \in E$ gilt.

Die **von** M **akzeptierte Sprache** ist

$$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}.$$

Von M akzeptierte Sprache:

Menge aller Wörter, die M vom Startzustand in einen Endzustand überführen.

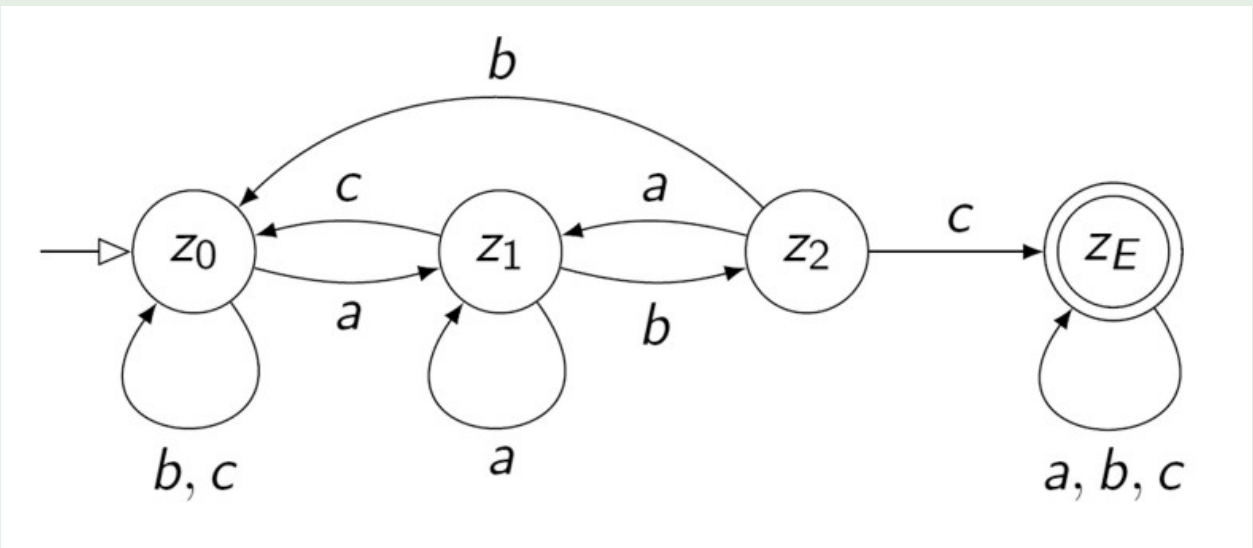
Akzeptierte Sprache eines DFA (II)

Beispiel 1

Sei $\Sigma = \{a, b, c\}$.

DFA, der die folgende Sprache L akzeptiert:

$$L = \{w \in \Sigma^* \mid w \text{ enthält } abc\}$$



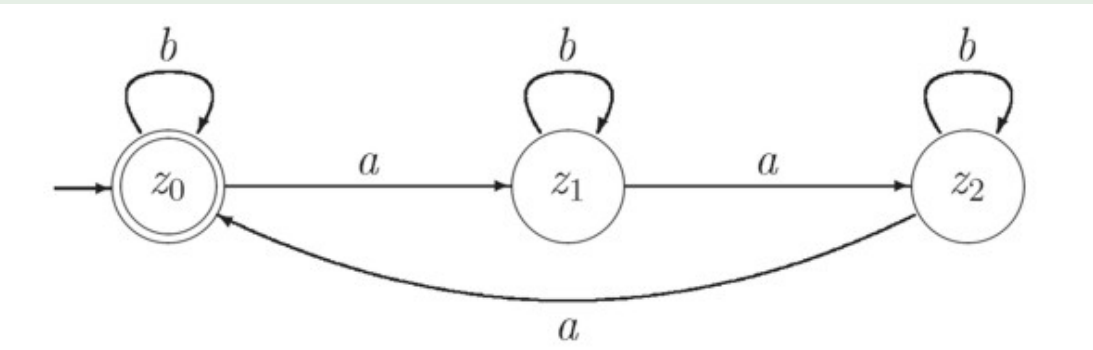
Akzeptierte Sprache eines DFA (III)

Beispiel 2

$M_{3a} = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_0\})$ mit

$$\begin{array}{lll} \delta(z_0, a) = z_1, & \delta(z_1, a) = z_2, & \delta(z_2, a) = z_0, \\ \delta(z_0, b) = z_0, & \delta(z_1, b) = z_1, & \delta(z_2, b) = z_2. \end{array}$$

Zustandsübergangsgraph



Akzeptierte Sprache eines DFA (IV)

Beispiel 2 (Fortsetzung)

Dann gilt z.B.

$$
\begin{aligned}
\hat{\delta}(z_0, \varepsilon) &= z_0 \in \{z_0\} \\
\hat{\delta}(z_0, abaa) &= \hat{\delta}(\delta(z_0, a), baa) \\
&= \hat{\delta}(z_1, baa) \\
&= \hat{\delta}(\delta(z_1, b), aa) \\
&= \hat{\delta}(z_1, aa) \\
&= \hat{\delta}(\delta(z_1, a), a) \\
&= \hat{\delta}(z_2, a) \\
&= \hat{\delta}(\delta(z_2, a), \varepsilon) \\
&= \hat{\delta}(z_0, \varepsilon) \\
&= z_0 \in \{z_0\}
\end{aligned}
$$

Akzeptierte Sprache eines DFA (V)

Beispiel 2 (Fortsetzung)

$$\begin{aligned}\hat{\delta}(z_0, bba) &= \hat{\delta}(z_0, ba)\\ &= \hat{\delta}(z_0, a)\\ &= z_1 \notin \{z_0\}\end{aligned}$$

Also: $\varepsilon \in L(M_{3a})$, $abaa \in L(M_{3a})$, $bba \notin L(M_{3a})$.

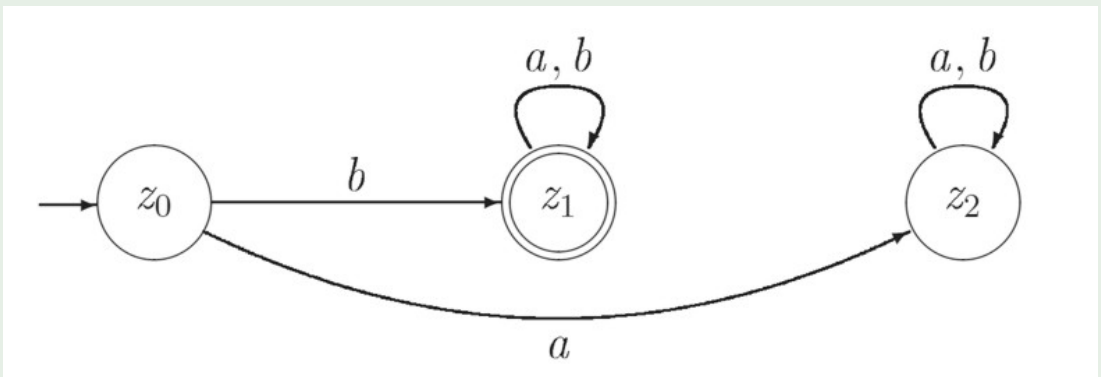
Akzeptierte Sprache eines DFA (VI)

Beispiel 3

DFA, der die folgende Sprache akzeptiert (mit $\Sigma = \{a, b\}$):

$$L_b = \{w \in \Sigma^* \mid w = bw' \text{ mit } w' \in \Sigma^*\}$$

Zustandsgraph:



Beachte: z_2 („Fehler-“) Zustand,
von dem aus kein akzeptierender Zustand mehr erreichbar ist.

Zusammenhang DFA und reguläre Grammatik (I)

Konstruktion: DFA $\rightsquigarrow$ reguläre Grammatik

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA.

Konstruiere Grammatik $G_M = (Z, \Sigma, P, z_0)$ mit Regeln:

$$P: \quad \begin{array}{ll} z \to az' & \text{für alle } z, z' \in Z,\ a \in \Sigma \text{ mit } \delta(z,a) = z', \\ z \to \varepsilon & \text{für alle } z \in E. \end{array}$$

G_M ist offenbar regulär.

Anschaulich:

$$\begin{array}{rcl} \text{Zustandsbezeichner } z_i \text{ von } M & \rightsquigarrow & \text{Variablensymbole in } G_M \\ \text{Zustandsübergänge} & \rightsquigarrow & \text{Regeln} \end{array}$$

Zusammenhang DFA und reguläre Grammatik (II)

Beispiel

Für den DFA $M_{3a} = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_0\})$ mit

$$\begin{aligned}
\delta(z_0, a) &= z_1 \\
\delta(z_0, b) &= z_0 \\
\delta(z_1, a) &= z_2 \\
\delta(z_1, b) &= z_1 \\
\delta(z_2, a) &= z_0 \\
\delta(z_2, b) &= z_2
\end{aligned}$$

ist $G_{M_{3a}} = (\{z_0, z_1, z_2\}, \{a, b\}, P, z_0)$ mit

$$\begin{aligned}
z_0 &\to az_1 \\
z_0 &\to bz_0 \\
z_1 &\to az_2 \\
z_1 &\to bz_1 \\
z_2 &\to az_0 \\
z_2 &\to bz_2 \\
z_0 &\to \varepsilon
\end{aligned}$$

Beobachtung: $G_{M_{3a}}$ ist die ursprüngliche Grammatik G_{3a} für L_{3a} (bis auf Bezeichnung der Variablen (hier: „z_i“)).

Es gilt also: $L(G_{M_{3a}}) = L_{3a} = L(M_{3a})$.

Zusammenhang DFA und reguläre Grammatik (III)

Allgemein gilt:

Satz 2.1

Für jeden DFA M ist $L(G_M) = L(M)$.

Direkt aus diesem Satz folgt insbesondere:

Satz 2.2

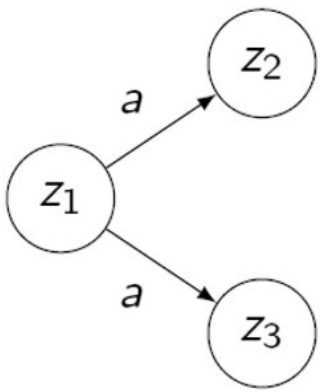
Jede von einem DFA akzeptierte Sprache ist regulär.

Nichtdeterministische endliche Automaten (NFA)

Idee nichtdeterministischer endlicher Automaten (I)

- Nichtdeterminismus: erlaube, dass ein Zustand für ein Eingabesymbol mehrere oder keine Folgezustände hat.

- **Ziel:** Vereinfachung/Unterspezifizierung.
  Führt zu kleineren oder besser verständlichen Automaten.

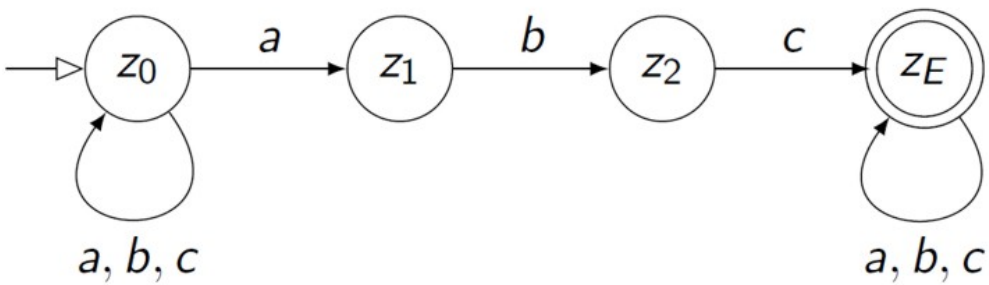


Idee nichtdeterministischer endlicher Automaten (II)

Nichtdeterministische Automaten: Zu jedem Paar

$$(z, a) \qquad \text{mit } z \text{ Zustand}, \quad a \text{ Eingabesymbol}$$

gibt es entweder **keinen**, **einen** oder **mehrere** Nachfolgezustände.



Nichtdeterministischer Automat

- wählt aus den möglichen Nachfolgerzuständen immer (nichtdeterministisch) einen aus.
- akzeptiert ein Wort, falls es bei „richtiger“ Auswahl vom Startzustand in einen Endzustand führt.

Nichtdeterministische endliche Automaten

Definition

Ein **nichtdeterministischer endlicher Automat (nondeterministic finite automaton, NFA)** ist ein 5-Tupel

$$M = (Z, \Sigma, \delta, z_0, E) \qquad \text{gegeben durch:}$$

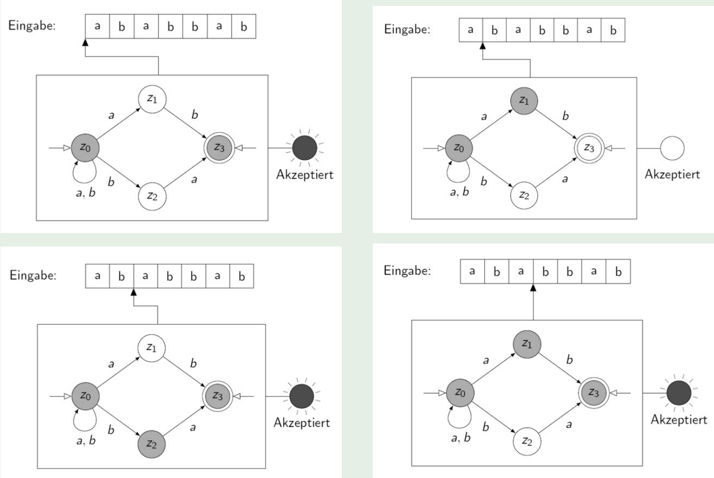
- Z, Σ, z_0, E wie bei DFA,
- eine totale Überführungsfunktion $\delta : Z \times \Sigma \to \wp(Z)$.

$\wp(Z)$ ist die Potenzmenge der Menge der Zustände:
δ bildet jedes Paar $(z, a) \in Z \times \Sigma$ auf eine Teilmenge von Z ab.
(die möglichen Folgezustände von z bei Eingabe von a)

Alternativ: (und equivalent) $\qquad \delta \subseteq (Z \times \Sigma) \times Z$ Relation

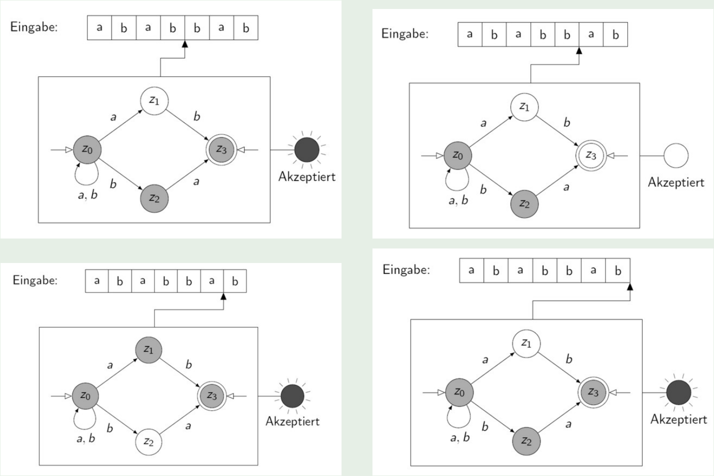
Ablauf eines NFA (I)

Beispiel



Ablauf eines NFA (II)

Beispiel (Fortsetzung)



Mehr-Schritt-Übergänge eines NFA (I)

Erweiterung von δ auf ganze Wörter $w \in \Sigma^*$:

Informell: für Zustandsmenge $Z' \subseteq Z$ ist $\hat{\delta}(Z', w)$ die Menge aller Zustände, die von Z' aus durch Abarbeitung von w erreicht werden können.

Definition (Mehr-Schritt-Übergänge)

Zu einem gegebenen NFA $M = (Z, \Sigma, \delta, z_0, E)$ definieren wir eine Funktion $\hat{\delta} : \wp(Z) \times \Sigma^* \to \wp(Z)$ induktiv wie folgt:

$$\begin{aligned} \hat{\delta}(Z', \varepsilon) &= Z' \quad \text{für } Z' \in \wp(Z), \\ \hat{\delta}(Z', aw) &= \bigcup_{z \in Z'} \hat{\delta}(\delta(z, a), w) \quad \text{für } Z' \in \wp(Z), a \in \Sigma, w \in \Sigma^*. \end{aligned}$$

mit $Z' \subseteq Z$, $w \in \Sigma^*$ und $a \in \Sigma$.

Akzeptierte Sprache eines NFA (I)

Definition

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein NFA.

M **akzeptiert** ein Wort $w \in \Sigma^*$, falls $\hat{\delta}(\{z_0\}, w) \cap E \neq \emptyset$ gilt.

Die **von** M **akzeptierte Sprache** ist:

$$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}.$$

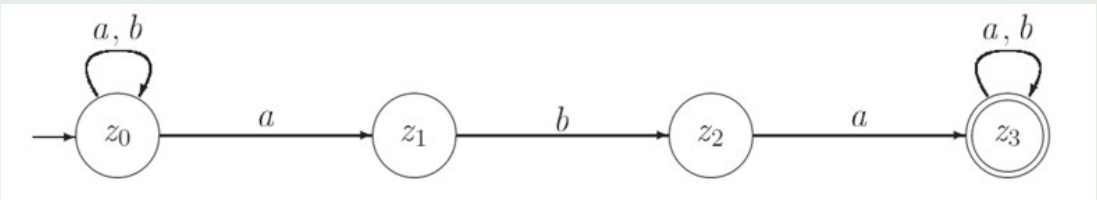
Von M akzeptierte Sprache:
Menge aller Wörter, die M vom Startzustand in einen Endzustand überführen. (Es kann für ein Wort mehrere Überführungspfade geben.)

Bemerkung: Jeder DFA ist ein (spezieller) NFA.
Die graphische Darstellung von NFA ist analog zu der von DFA.

Akzeptierte Sprache eines NFA (II)

Beispiel

Der NFA M_{aba} sei gegeben durch



Es ist z. B. $baabab \in L(M_{aba})$, denn:

$$\begin{aligned}
\hat{\delta}(\{z_0\}, baabab) &= \hat{\delta}(\{z_0\}, aabab) \\
&= \hat{\delta}(\{z_0, z_1\}, abab) \\
&= \hat{\delta}(\{z_0, z_1\}, bab) \\
&= \hat{\delta}(\{z_0, z_2\}, ab) \\
&= \hat{\delta}(\{z_0, z_1, z_3\}, b)
\end{aligned}$$

Akzeptierte Sprache eines NFA (III)

Beispiel (Fortsetzung)

$$\begin{aligned}\hat{\delta}(\{z_0\}, baabab) &= \cdots \\ &= \hat{\delta}(\{z_0, z_2, z_3\}, \varepsilon) \\ &= \{z_0, z_2, z_3\}\end{aligned}$$

und $\{z_0, z_2, z_3\} \cap E \neq \emptyset$.

Allgemeiner gilt (ohne Beweis):

$$L(M_{aba}) = \{w \in \{a, b\}^* \mid aba \text{ ist Teilwort von w}\}.$$

Zusammenhang NFA und reguläre Grammatik (I)

- **Zuvor:** Konstruktion DFA $\rightsquigarrow$ reguläre Grammatik
- **Jetzt:** „Umkehrung“ der Konstruktion liefert:

$$\text{reguläre Grammatik} \rightsquigarrow \text{NFA}$$

Zusammenhang NFA und reguläre Grammatik (II)

Konstruktion: Sei $G = (V, \Sigma, P, S)$ eine reguläre Grammatik.

Der NFA M_G ist gegeben durch $M_G = (V', \Sigma, \delta, S, E)$ mit

- Zustände von M_G:

$$V' = \begin{cases} V \cup \{X\} \text{ mit } X \notin V, & \text{falls P eine Regel der Form} \\ & A \to a \text{ enthält} \\ V & \text{sonst,} \end{cases}$$

- Zustandsübergänge von M_G:

  $B \in \delta(A, a)$ für Regel $A \to aB$ in P (für $B \in V$),

  $X \in \delta(A, a)$ für Regel $A \to a$ in P,

- Endzustände von M_G:

$$E = \begin{cases} \{A \in V \mid P \text{ enthält } A \to \varepsilon\} \cup \{X\}, & \text{falls } X \in V' \\ \{A \in V \mid P \text{ enthält } A \to \varepsilon\} & \text{sonst.} \end{cases}$$

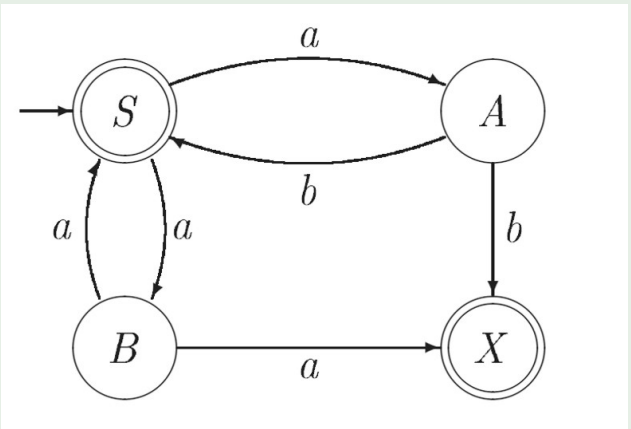
Zusammenhang NFA und reguläre Grammatik (III)

Beispiel

Sei $G = (\{A, B, S\}, \{a, b\}, P, S)$ mit den Regeln

$$S \to aA \mid aB \mid \varepsilon,$$
$$A \to bS \mid b,$$
$$B \to aS \mid a.$$

M_G ist dann der NFA



Zusammenhang NFA und reguläre Grammatik (IV)

Allgemein gilt:

Satz 2.3

Für jede reguläre Grammatik G ist $L(M_G) = L(G)$.

Daraus folgt insbesondere:

Satz 2.4

Zu jeder regulären Sprache L gibt es einen NFA M mit $L(M) = L$.

Wir haben nun folgende Konstruktionen beschrieben:

$$\text{DFA} \rightsquigarrow \text{reguläre Grammatik} \rightsquigarrow \text{NFA}$$

Zusammenhang NFA und DFA (I)

Jetzt: Konstruktion NFA $N \rightsquigarrow$ DFA M_N

Idee des Verfahrens:

- (potenzielle) Zustände von M_N:
  jede Teilmenge der Zustandsmenge des NFA N
- konstruiere die Übergänge
- eliminiere alle nicht erforderlichen Zustände des DFA M_N:
  nicht erreichbare Teilmengen / Zustände von M_N

Beachte:
Falls N n Zustände hat, hat M_N im schlimmsten Fall 2^n Zustände.

Zusammenhang NFA und DFA (II)

Potenzmenge-Konstruktion:

Es sei $N = (Z, \Sigma, \delta, z_0, E)$ NFA.

Dann ist $M_N = (\wp(Z), \Sigma, \delta', \{z_0\}, E')$ mit

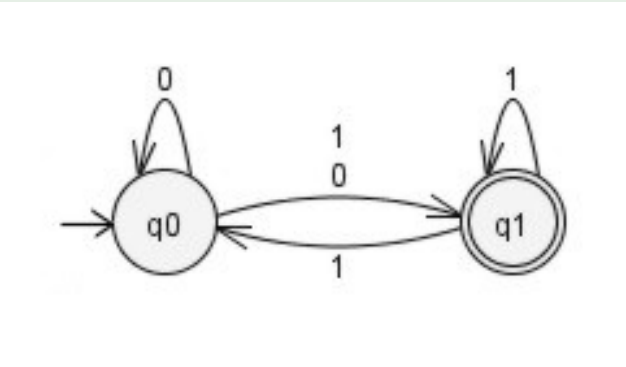
- $\delta'(Z', a) = \bigcup_{z \in Z'} \delta(z, a)$ für alle $Z' \in \wp(Z)$,
  - betrachte alle in Z' enthaltenen Zustände z,
  - untersuche welche Zustände N auf Eingabe a von z aus erreicht
  - nimm die Vereinigung aller dieser Zustände.
- $E' = \{Z' \in \wp(Z) \mid Z' \cap E \neq \emptyset\}$

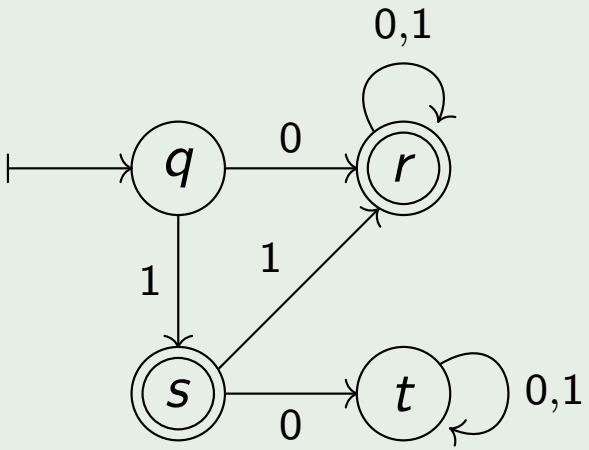
  Menge der Endzustände von M_N:

  alle Teilmengen von Z, die mindestens einen Endzustand aus E enthalten.

Zusammenhang NFA und DFA (III)

Beispiel





δ'	q	r	s	t
	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$	$\emptyset$
0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\emptyset$	$\emptyset$
1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\emptyset$
final?	nein	ja	ja	nein

Zusammenhang NFA und DFA (IV)

Es gelten:

Satz 2.5

Für jeden NFA N ist $L(M_N) = L(N)$.

Satz 2.6

Zu jedem NFA M gibt es einen DFA M' mit $L(M') = L(M)$.

Reguläre Grammatiken, DFA und NFA (I)

Bisher: drei Formalismen für reguläre Sprachen:

reguläre Grammatiken, DFA und NFA.

Vor- und Nachteile der Formalismen:

- Reguläre Grammatiken
  - schaffen Verbindung zur Chomsky-Hierarchie
  - werden zur Erzeugung von Sprachen eingesetzt
  - nichtdeterministisch $\rightsquigarrow$ schwieriger zu entscheiden, ob ein bestimmtes Wort erzeugt wird
- NFA
  - erlauben oft kleine, kompakte Darstellungen von Sprachen
  - haben intuitive, graphische Notation
  - nichtdeterministisch $\rightsquigarrow$ schwieriger zu entscheiden, ob ein bestimmtes Wort akzeptiert wird

Reguläre Grammatiken, DFA und NFA (II)

- DFA
  - können gegenüber äquivalenten NFA exponentiell größer werden
  - erlauben eine effiziente Lösung des Wortproblems (einfach Übergänge des Automaten durchführen und überprüfen, ob ein Endzustand erreicht wird)

Alle Modelle:

relativ viel Schreibaufwand und Platz für die Notation.

Gesucht: kompaktere syntaktische Repräsentation

$\rightsquigarrow$ **reguläre Ausdrücke**.

Reguläre Ausdrücke

Syntax regulärer Ausdrücke

Definition (induktiv)

Menge Reg_Σ der **regulären Ausdrücke** über Σ
(für ein gegebenes Alphabet Σ):

1. $\emptyset \in \mathrm{Reg}_\Sigma$, $\varepsilon \in \mathrm{Reg}_\Sigma$ und $a \in \mathrm{Reg}_\Sigma$ für alle $a \in \Sigma$.
2. Falls $r, s \in \mathrm{Reg}_\Sigma$, dann $(r \cdot s), (r + s), r^* \in \mathrm{Reg}_\Sigma$.

Bemerkungen:

- Schreibe: $(r|s)$ statt $(r + s)$ und (rs) statt $(r \cdot s)$
- Anschaulich: $(r|s)$ bzw. $(r + s)$ bedeutet Alternative,
  $(r \cdot s)$ bzw. (rs) bedeutet Verkettung und
  $(\cdots)^*$ bedeutet „beliebig oft“.

Beispiele

$(ab|ba)$
$(ab|ba)^*$

$(ab|ba)c^*$
$(a|b|c)^*abc(a|b|c)^*$

Durch regulärer Ausdrücke beschriebene Sprache (I)

Nach Syntax jetzt **Semantik** regulärer Ausdrücke.

(d.h. Bedeutung: welche Sprache beschreibt ein regulärer Ausdruck?)

Definition

Für $r \in \mathrm{Reg}_\Sigma$ ist die **durch** r **beschriebene Sprache** $L(r)$ rekursiv definiert wie folgt:

1. $L(\emptyset) = \emptyset, \quad L(\varepsilon) = \{\varepsilon\}, \quad L(a) = \{a\}$ für $a \in \Sigma$.
2. $$\begin{aligned} L(r \cdot s) &= L(r) \circ L(s), \\ L(r + s) &= L(r) \cup L(s), \\ L(r^*) &= L(r)^*. \end{aligned}$$

Durch regulärer Ausdrücke beschriebene Sprache (II)

Beispiele für reguläre Ausdrücke über dem Alphabet $\Sigma = \{a, b\}$

- Beispiel 1: Sprache aller Wörter, die mit a beginnen und mit bb enden.

$$\alpha = a(a|b)^*bb$$

- Beispiel 2: Sprache aller Wörter, die das Teilwort aba enthalten.

$$\alpha = (a|b)^*aba(a|b)^*$$

- Beispiel 3: Sprache aller Wörter, die eine gerade Anzahl des Zeichens a enthalten.

$$\alpha = (b^*ab^*a)^*b^*$$

Durch regulärer Ausdrücke beschriebene Sprache (III)

Konventionen: (zur Klammereinsparung)

- Bindungsregel: „$\cdot$ vor $+$“.
- $r \cdot s \cdot t$ statt $(r \cdot s) \cdot t$ oder $r \cdot (s \cdot t)$ (da „$\cdot$“) assoziativ)
- analog für $+$ bzw. $|$.

Beispiele

Sei $\Sigma = \{a, b\}$. Dann gilt:

- $L(a|b) = \Sigma$
- $L((a|b)^*) = \Sigma^*$
- $L((ab)^*a) = \{(ab)^n a \in \Sigma^* \mid n \in \mathbb{N}\}$
- $L(a(a|b)^*b|b) = \{w \in \Sigma^* \mid w = aw'b \text{ mit } w' \in \Sigma^* \text{ oder } w = b\}$

Reguläre Sprache und regulärer Ausdruck

Satz 2.7 (Satz von Kleene)

Sei Σ ein Alphabet. Eine Sprache L über Σ ist genau dann regulär, wenn es einen regulären Ausdruck $r \in \mathrm{Reg}_\Sigma$ gibt mit $L(r) = L$.

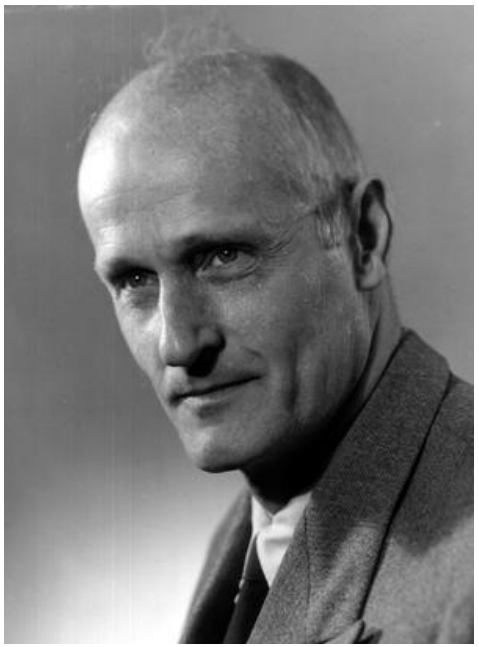
Bemerkung: der Beweis liefert zwei Konstruktionen:

$$\text{DFA} \rightsquigarrow \text{regulärer Ausdruck} \rightsquigarrow \text{DFA}$$

Stephen Cole Kleene

Amerikanischer Mathematiker

- einer der Begründer der Rekursionstheorie
- wichtige Beiträge zur Theoretischen Informatik
- Theorie der berechenbaren Funktionen



Stephen Cole Kleene (1909–1994)
Quelle: www.library.wisc.edu

Bekannte Entdeckungen:

- Erfinder der regulären Ausdrücke
- Kleene'scher Fixpunktsatz
- Kleene'scher Rekursionssatz

Außerdem nach Kleene benannt:

Kleene Stern, Kleene Algebra, Kleene Hierarchie

Zusammenhang reguläre Grammatik, DFA, NFA, regulärer Ausdruck

Satz 2.8

Sei L eine Sprache. Die folgenden Aussagen sind äquivalent:

a) L wird von einer regulären Grammatik erzeugt (d.h. L ist regulär).

b) L wird von einem DFA akzeptiert.

c) L wird von einem NFA akzeptiert.

d) L wird durch einen regulären Ausdruck beschrieben.

Zusammenfassung