

Vorlesung
Grundlagen der Theoretischen Informatik /
Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik



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*

Endliche Automaten

- 1 Determinierte endliche Automaten (DEAs)
- 2 Indeterminierte endliche Automaten (NDEAs)
- 3 Automaten mit ε -Kanten
- 4 Endliche Automaten \equiv Typ-3-Sprachen
- 5 Pumping Lemma
- 6 Wortprobleme
- 7 Rational = Regulär

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear, regulär**)
- **Determinierte** und **indeterminierte** endliche Automaten sind äquivalent
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- Es gibt Algorithmen, die **Probleme über endlichen Automaten** bzw. Typ-3-Sprachen lösen.
- Typ-3-Sprachen sind genau die, die durch **reguläre Ausdrücke** beschrieben werden können.

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear, regulär**)
- **Determinierte** und **indeterminierte** endliche Automaten sind äquivalent
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- Es gibt Algorithmen, die **Probleme über endlichen Automaten** bzw. Typ-3-Sprachen lösen.
- Typ-3-Sprachen sind genau die, die durch **reguläre Ausdrücke** beschrieben werden können.

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear**, **regulär**)
- **Determinierte und indeterminierte endliche Automaten sind äquivalent**
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- Es gibt Algorithmen, die **Probleme über endlichen Automaten** bzw. Typ-3-Sprachen lösen.
- Typ-3-Sprachen sind genau die, die durch **reguläre Ausdrücke** beschrieben werden können.

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear**, **regulär**)
- **Determinierte** und **indeterminierte** endliche Automaten sind äquivalent
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- Es gibt Algorithmen, die **Probleme über endlichen Automaten** bzw. Typ-3-Sprachen lösen.
- Typ-3-Sprachen sind genau die, die durch **reguläre Ausdrücke** beschrieben werden können.

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear**, **regulär**)
- **Determinierte** und **indeterminierte** endliche Automaten sind äquivalent
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- **Es gibt Algorithmen, die Probleme über endlichen Automaten bzw. Typ-3-Sprachen lösen.**
- Typ-3-Sprachen sind genau die, die durch **reguläre Ausdrücke** beschrieben werden können.

Inhalt von Teil III

- Vereinfachtes Modell eines Computers: **endlicher Automat**
- Die von endlichen Automaten erkannten „**rationalen**“ Sprachen sind genau die Typ-3-Sprachen (**rechtslinear**, **regulär**)
- **Determinierte** und **indeterminierte** endliche Automaten sind äquivalent
- **Pumping Lemma** erlaubt, eine Sprache als nicht rational nachzuweisen.
- Es gibt Algorithmen, die **Probleme über endlichen Automaten** bzw. Typ-3-Sprachen lösen.
- **Typ-3-Sprachen sind genau die, die durch reguläre Ausdrücke beschrieben werden können.**

Endliche Automaten

- 1 Determinierte endliche Automaten (DEAs)
- 2 Indeterminierte endliche Automaten (NDEAs)
- 3 Automaten mit ε -Kanten
- 4 Endliche Automaten \equiv Typ-3-Sprachen
- 5 Pumping Lemma
- 6 Wortprobleme
- 7 Rational = Regulär

Endliche Automaten

1 **Determinierte endliche Automaten (DEAs)**

2 Indeterminierte endliche Automaten (NDEAs)

3 Automaten mit ε -Kanten

4 Endliche Automaten \equiv Typ-3-Sprachen

5 Pumping Lemma

6 Wortprobleme

7 Rational = Regulär

Beispiel 11.1

Die Sprache

$$L = \{aa\}\{ab\}^*\{c\}$$

ist **regulär**.

Denn sie wird (z. B.) erzeugt von der **rechtslinearen** Grammatik

$$G = (\{S, A\}, \{a, b, c\}, R, S),$$

mit Regelmenge R :

$$S \rightarrow aaA$$

$$A \rightarrow abA \mid c$$

Beispiel 11.1

Die Sprache

$$L = \{aa\}\{ab\}^*\{c\}$$

ist **regulär**.

Denn sie wird (z. B.) erzeugt von der **rechtslinearen** Grammatik

$$G = (\{S, A\}, \{a, b, c\}, R, S),$$

mit Regelmenge R :

$$S \rightarrow aaA$$

$$A \rightarrow abA \mid c$$

Beispiel 11.2

Die Sprache aller durch 3 teilbaren Dezimalzahlen ist regulär.

Eine erzeugende Grammatik ist

$$G = (\{S, S_0, S_1, S_2\}, \{0, \dots, 9\}, R, S)$$

mit der Regelmenge R :

$$\begin{aligned} S &\rightarrow 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid 0 \\ S_0 &\rightarrow 0S_0 \mid 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid \varepsilon \\ S_1 &\rightarrow 0S_1 \mid 3S_1 \mid 6S_1 \mid 9S_1 \mid 1S_2 \mid 4S_2 \mid 7S_2 \mid 2S_0 \mid 5S_0 \mid 8S_0 \\ S_2 &\rightarrow 0S_2 \mid 3S_2 \mid 6S_2 \mid 9S_2 \mid 1S_0 \mid 4S_0 \mid 7S_0 \mid 2S_1 \mid 5S_1 \mid 8S_1 \end{aligned}$$

Ohne das ε in der zweiten Regel wäre nur die "0" als Terminalwort herleitbar.

Beispiel

Beispiel 11.2

Die Sprache aller durch 3 teilbaren Dezimalzahlen ist regulär.

Eine erzeugende Grammatik ist

$$G = (\{S, S_0, S_1, S_2\}, \{0, \dots, 9\}, R, S)$$

mit der Regelmenge R :

$$\begin{aligned} S &\rightarrow 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid 0 \\ S_0 &\rightarrow 0S_0 \mid 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid \varepsilon \\ S_1 &\rightarrow 0S_1 \mid 3S_1 \mid 6S_1 \mid 9S_1 \mid 1S_2 \mid 4S_2 \mid 7S_2 \mid 2S_0 \mid 5S_0 \mid 8S_0 \\ S_2 &\rightarrow 0S_2 \mid 3S_2 \mid 6S_2 \mid 9S_2 \mid 1S_0 \mid 4S_0 \mid 7S_0 \mid 2S_1 \mid 5S_1 \mid 8S_1 \end{aligned}$$

Ohne das ε in der zweiten Regel wäre nur die "0" als Terminalwort herleitbar.

Beispiel

Beispiel 11.2

Die Sprache aller durch 3 teilbaren Dezimalzahlen ist regulär.

Eine erzeugende Grammatik ist

$$G = (\{S, S_0, S_1, S_2\}, \{0, \dots, 9\}, R, S)$$

mit der Regelmenge R :

$$\begin{aligned} S &\rightarrow 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid 0 \\ S_0 &\rightarrow 0S_0 \mid 3S_0 \mid 6S_0 \mid 9S_0 \mid 1S_1 \mid 4S_1 \mid 7S_1 \mid 2S_2 \mid 5S_2 \mid 8S_2 \mid \varepsilon \\ S_1 &\rightarrow 0S_1 \mid 3S_1 \mid 6S_1 \mid 9S_1 \mid 1S_2 \mid 4S_2 \mid 7S_2 \mid 2S_0 \mid 5S_0 \mid 8S_0 \\ S_2 &\rightarrow 0S_2 \mid 3S_2 \mid 6S_2 \mid 9S_2 \mid 1S_0 \mid 4S_0 \mid 7S_0 \mid 2S_1 \mid 5S_1 \mid 8S_1 \end{aligned}$$

Ohne das ε in der zweiten Regel wäre nur die "0" als Terminalwort herleitbar.

Grammatik vs. Automat

Grammatik: erzeugt Wörter

Automat: analysiert / erkennt Wörter

beide: beschreiben / definieren eine Sprachen

Grammatik vs. Automat

Grammatik: erzeugt Wörter

Automat: analysiert / erkennt Wörter

beide: beschreiben / definieren eine Sprachen

Grammatik vs. Automat

Grammatik: erzeugt Wörter

Automat: analysiert / erkennt Wörter

beide: beschreiben / definieren eine Sprachen

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- **Endlich viele mögliche interne Zustände, immer einer davon ist der aktuelle Zustand**
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand, in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist, ist w **akzeptiert** als Element von L , sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- **Automat beginnt in einem initialen Zustand.**
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat: Informell

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- Automat **stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat

- Ein endlicher Automat testet, ob ein gegebenes $w \in \Sigma^*$ in einer Sprache L liegt.
- **Lesekopf** erlaubt w zu lesen.
Bewegt sich nur von links nach rechts.
- Endlich viele mögliche **interne Zustände**,
immer einer davon ist der aktuelle Zustand
- Automat beginnt in einem **initialen Zustand**.
- Bei jedem gelesenen Buchstaben Übergang zu neuem aktuellen Zustand,
in Abhängigkeit vom Buchstaben und dem alten Zustand
- Wenn am Ende von w ein **finaler Zustand** erreicht ist,
ist w **akzeptiert** als Element von L ,
sonst nicht.
- **Automat stoppt** (auf jeden Fall) nach $|w|$ Schritten

Endlicher Automat: Computer mit begrenztem Speicher

- Kann vom Band nur lesen
⇒ kein externer Speicher
- Speichert nur den aktuellen Zustand (\approx Programmzähler)
⇒ stark begrenzter interner Speicher

Endlicher Automat: Computer mit begrenztem Speicher

- Kann vom Band nur lesen
⇒ kein externer Speicher
- Speichert nur den aktuellen Zustand (\approx Programmzähler)
⇒ stark begrenzter interner Speicher

Darstellung als Graph

- ein **Knoten** für jeden **Zustand**
- **Kanten** beschreiben **Zustandsänderungen**, sind mit Buchstaben beschriftet
- **initiale** Zustände sind mit einem **Pfeil** gekennzeichnet
- **finale Zustände** mit einem **doppelten Kreis**

Darstellung als Graph

- ein **Knoten** für jeden **Zustand**
- **Kanten** beschreiben **Zustandsänderungen**, sind mit **Buchstaben** beschriftet
- **initiale** Zustände sind mit einem **Pfeil** gekennzeichnet
- **finale Zustände** mit einem **doppelten Kreis**

Darstellung als Graph

- ein **Knoten** für jeden **Zustand**
- **Kanten** beschreiben **Zustandsänderungen**, sind mit Buchstaben beschriftet
- **initiale Zustände** sind mit einem **Pfeil** gekennzeichnet
- **finale Zustände** mit einem **doppelten Kreis**

Darstellung als Graph

- ein **Knoten** für jeden **Zustand**
- **Kanten** beschreiben **Zustandsänderungen**, sind mit Buchstaben beschriftet
- **initiale** Zustände sind mit einem **Pfeil** gekennzeichnet
- **finale Zustände** mit einem **doppelten Kreis**

Endlicher Automat: Darstellung als Graph

Beispiel 11.3 (Sprache $\{w \mid \#_a(w) \text{ gerade}\} \subset \{a,b\}^*$)

Der folgende endliche Automat erkennt die Sprache

$$\{w \mid \#_a(w) \text{ gerade}\} \quad \text{über} \quad \Sigma = \{a,b\}$$

der Wörter mit gerader Anzahl von „a“s

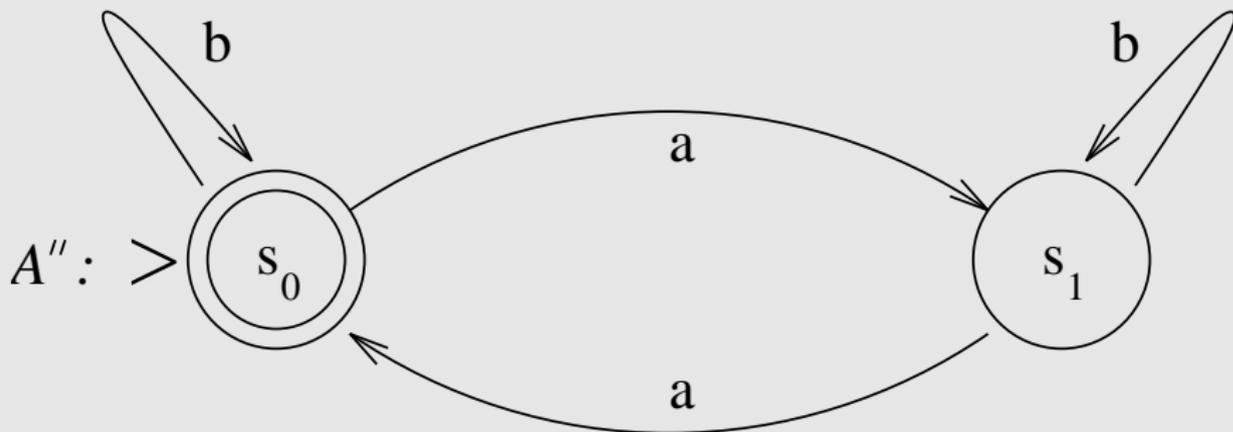
Endlicher Automat: Darstellung als Graph

Beispiel 11.3 (Sprache $\{w \mid \#_a(w) \text{ gerade}\} \subset \{a,b\}^*$)

Der folgende endliche Automat erkennt die Sprache

$$\{w \mid \#_a(w) \text{ gerade}\} \quad \text{über} \quad \Sigma = \{a,b\}$$

der Wörter mit gerader Anzahl von „a“s



Endlicher Automat: Definition

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat (e.a., finite automaton)** ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine endliche Menge von **Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta: K \times \Sigma \rightarrow K$ die totale(!) **Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat (e.a., finite automaton)** ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine **endliche Menge von Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta : K \times \Sigma \rightarrow K$ die totale(!) **Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Endlicher Automat: Definition

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat (e.a., finite automaton)** ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine endliche Menge von **Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta : K \times \Sigma \rightarrow K$ die totale(!) **Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Endlicher Automat: Definition

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat (e.a., finite automaton)** ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine endliche Menge von **Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta : K \times \Sigma \rightarrow K$ die **totale(!) Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Endlicher Automat: Definition

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat** (e.a., **finite automaton**) ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine endliche Menge von **Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta : K \times \Sigma \rightarrow K$ die totale(!) **Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Definition 11.4 (Endlicher Automat)

Ein **endlicher Automat (e.a., finite automaton)** ist ein Tupel

$$\mathcal{A} = (K, \Sigma, \delta, s_0, F)$$

Dabei ist

- K eine endliche Menge von **Zuständen**
- Σ ein **endliches Alphabet**
(aus dessen Buchstaben die Eingabewörter bestehen können)
- $\delta : K \times \Sigma \rightarrow K$ die totale(!) **Übergangsfunktion**
- $s_0 \in K$ der **Startzustand**
- $F \subseteq K$ die Menge der **finalen Zustände**

Bedeutung der Übergangsfunktion

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

bedeutet:

- Wenn der Automat im Zustand q ist
- und ein a liest,
- dann geht in den Zustand q' über.

Bedeutung der Übergangsfunktion

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

bedeutet:

- Wenn der Automat im Zustand q ist
- **und ein a liest,**
- dann geht in den Zustand q' über.

Bedeutung der Übergangsfunktion

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

bedeutet:

- Wenn der Automat im Zustand q ist
- und ein a liest,
- dann geht in den Zustand q' über.

Definition 11.5 (Erweiterung von δ zu δ^*)

$$\delta^* : K \times \Sigma^* \rightarrow K$$

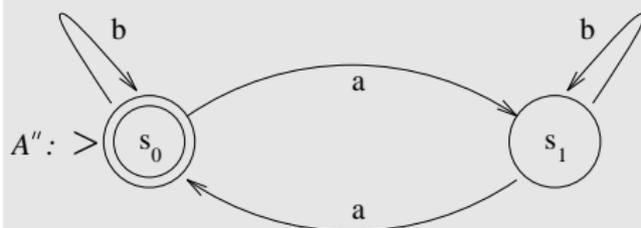
ist strukturell rekursiv über Σ^* definiert durch:

$$\delta^*(q, \varepsilon) := q$$

$$\delta^*(q, wa) := \delta(\delta^*(q, w), a)$$

Endlicher Automat: Beispiel

Beispiel 11.6



Dieser Automat akzeptiert die Sprache

$$\{w \mid \#_a(w) \text{ gerade}\} \subset \{a,b\}^*$$

(s. Bsp. 11.3).

Formal hat er die Form:

$$\mathcal{A} = (\{s_0, s_1\}, \{a, b\}, \delta, s_0, \{s_0\})$$

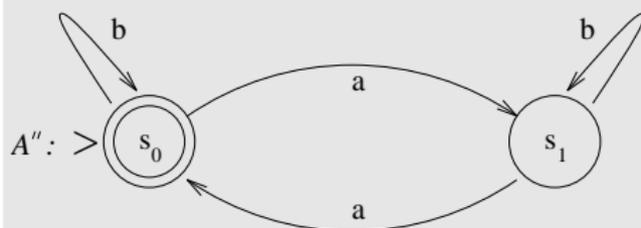
mit

$$\delta(s_0, a) = s_1 \quad \delta(s_1, a) = s_0$$

$$\delta(s_0, b) = s_0 \quad \delta(s_1, b) = s_1$$

Endlicher Automat: Beispiel

Beispiel 11.6



Dieser Automat akzeptiert die Sprache

$$\{w \mid \#_a(w) \text{ gerade}\} \subset \{a,b\}^*$$

(s. Bsp. 11.3).

Formal hat er die Form:

$$\mathcal{A} = (\{s_0, s_1\}, \{a, b\}, \delta, s_0, \{s_0\})$$

mit

$$\delta(s_0, a) = s_1 \quad \delta(s_1, a) = s_0$$

$$\delta(s_0, b) = s_0 \quad \delta(s_1, b) = s_1$$

Endlicher Automat: Übergangsfunktion

Beispiel 11.7 (Beispiel für δ^*)

$$\begin{aligned}\delta^*(s_0, aab) &= \delta(\delta^*(s_0, aa), b) \\ &= \delta(\delta(\delta^*(s_0, a), a), b) \\ &= \delta(\delta(\delta(\delta^*(s_0, \varepsilon), a), a), a), b) \\ &= \delta(\delta(\delta(s_0, a), a), b) \\ &= \delta(\delta(s_1, a), b) \\ &= \delta(s_0, b) \\ &= s_0\end{aligned}$$

Endlicher Automat: Akzeptierte Sprache

Definition 11.8 (Von einem endlichen Automaten akzeptierte Sprache)

Die von einem Automaten \mathcal{A} **akzeptierte** Sprache, ist definiert als

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \delta^*(s_0, w) \in F\}$$

Definition 11.9 (Von endlichen Automaten akzeptierte Sprachen)

Die Menge

$$\mathbf{RAT} := \{L \mid \text{es gibt einen endlichen Automaten } \mathcal{A} \text{ mit } L = L(\mathcal{A})\}$$

der von endlichen Automaten akzeptierten Sprachen
heißt Menge der **rationalen** Sprachen

Wir zeigen demnächst:

RAT = Menge der regulären Sprachen

Endlicher Automat: Akzeptierte Sprache

Definition 11.8 (Von einem endlichen Automaten akzeptierte Sprache)

Die von einem Automaten \mathcal{A} **akzeptierte** Sprache, ist definiert als

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \delta^*(s_0, w) \in F\}$$

Definition 11.9 (Von endlichen Automaten akzeptierte Sprachen)

Die Menge

$$\mathbf{RAT} := \{L \mid \text{es gibt einen endlichen Automaten } \mathcal{A} \text{ mit } L = L(\mathcal{A})\}$$

der von endlichen Automaten akzeptierten Sprachen
heißt Menge der **rationalen** Sprachen

Wir zeigen demnächst:

RAT = Menge der regulären Sprachen

Endlicher Automat: Akzeptierte Sprache

Definition 11.8 (Von einem endlichen Automaten akzeptierte Sprache)

Die von einem Automaten \mathcal{A} **akzeptierte** Sprache, ist definiert als

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \delta^*(s_0, w) \in F\}$$

Definition 11.9 (Von endlichen Automaten akzeptierte Sprachen)

Die Menge

$$\mathbf{RAT} := \{L \mid \text{es gibt einen endlichen Automaten } \mathcal{A} \text{ mit } L = L(\mathcal{A})\}$$

der von endlichen Automaten akzeptierten Sprachen

heißt Menge der **rationalen** Sprachen

Wir zeigen demnächst:

RAT = Menge der regulären Sprachen

Endlicher Automat: Akzeptierte Sprache

Definition 11.8 (Von einem endlichen Automaten akzeptierte Sprache)

Die von einem Automaten \mathcal{A} **akzeptierte** Sprache, ist definiert als

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \delta^*(s_0, w) \in F\}$$

Definition 11.9 (Von endlichen Automaten akzeptierte Sprachen)

Die Menge

$$\mathbf{RAT} := \{L \mid \text{es gibt einen endlichen Automaten } \mathcal{A} \text{ mit } L = L(\mathcal{A})\}$$

der von endlichen Automaten akzeptierten Sprachen
heißt Menge der **rationalen** Sprachen

Wir zeigen demnächst:

RAT = Menge der regulären Sprachen

Endlicher Automat: Akzeptierte Sprache

Definition 11.8 (Von einem endlichen Automaten akzeptierte Sprache)

Die von einem Automaten \mathcal{A} **akzeptierte** Sprache, ist definiert als

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \delta^*(s_0, w) \in F\}$$

Definition 11.9 (Von endlichen Automaten akzeptierte Sprachen)

Die Menge

$$\mathbf{RAT} := \{L \mid \text{es gibt einen endlichen Automaten } \mathcal{A} \text{ mit } L = L(\mathcal{A})\}$$

der von endlichen Automaten akzeptierten Sprachen
heißt Menge der **rationalen** Sprachen

Wir zeigen demnächst:

RAT = Menge der regulären Sprachen

Endliche Automaten: UML State Charts

UML State Charts

UML State Charts sind eine (erweiterte) Form endlicher Automaten

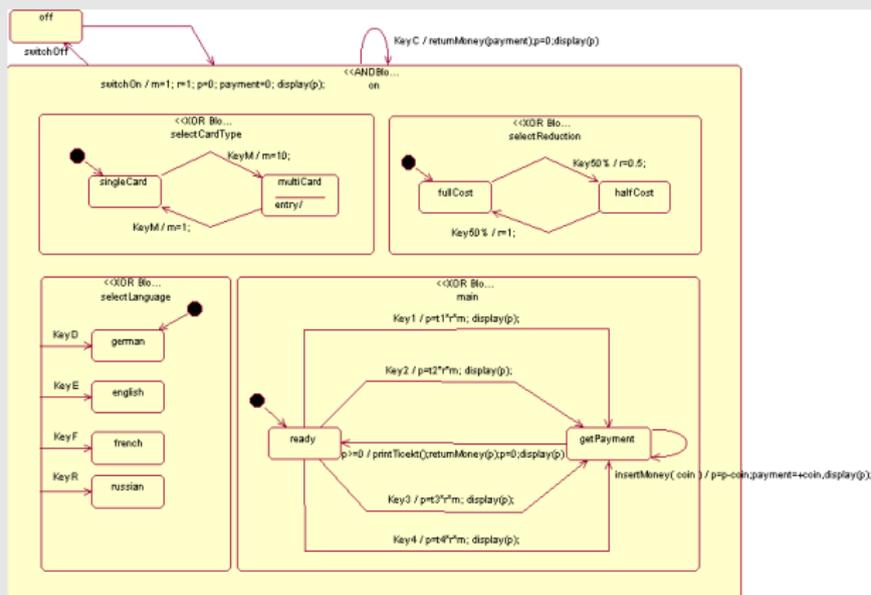
Beispiel 11.10

Endliche Automaten: UML State Charts

UML State Charts

UML State Charts sind eine (erweiterte) Form endlicher Automaten

Beispiel 11.10



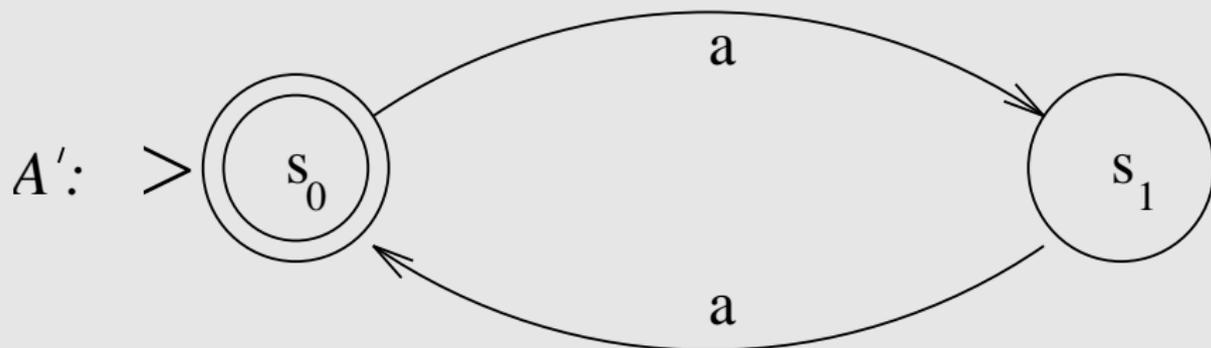
Beispiel 11.11

Die Sprache aller Wörter mit gerader Anzahl von a
über dem (kleineren) Alphabet $\Sigma = \{a\}$
wird akzeptiert von:

Endliche Automaten: Weitere Beispiele

Beispiel 11.11

Die Sprache aller Wörter mit gerader Anzahl von a
über dem (kleineren) Alphabet $\Sigma = \{a\}$
wird akzeptiert von:



Beispiel 11.12

Die Sprache

$$L = \{w \in \{0, 1\}^* \mid w \text{ enthält genau zwei Einsen}\}$$

wird akzeptiert von dem folgenden endlichen Automaten:

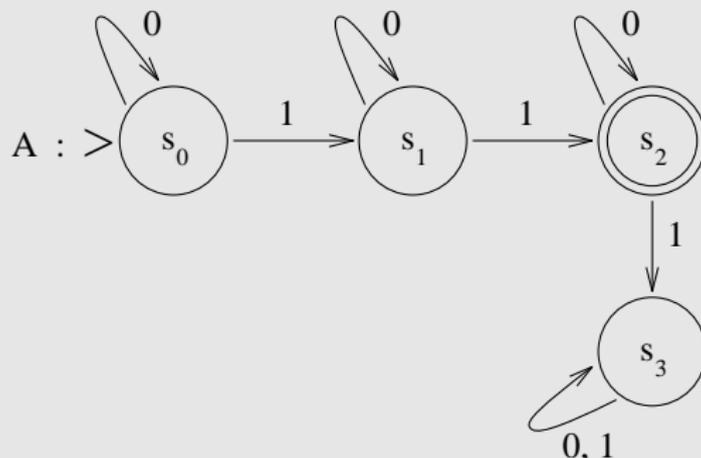
Endliche Automaten: Weitere Beispiele

Beispiel 11.12

Die Sprache

$$L = \{w \in \{0,1\}^* \mid w \text{ enthält genau zwei Einsen}\}$$

wird akzeptiert von dem folgenden endlichen Automaten:



Beispiel 11.13

Die Sprache aller durch 3 teilbaren Dezimalzahlen wird akzeptiert durch:

Endliche Automaten: Weitere Beispiele

Beispiel 11.13

Die Sprache aller durch 3 teilbaren Dezimalzahlen wird akzeptiert durch:

