

## Reguläre Sprachen: Formalismen

Prof. Dr. David Sabel

LFE Theoretische Informatik



Letzte Änderung der Folien: 4. Juni 2019

## Inhaltsübersicht

- Deterministische endliche Automaten
- Nichtdeterministische endliche Automaten
- Reguläre Ausdrücke
- Äquivalenz der Formalismen

## Reguläre Sprachen

### Wiederholung:

- Eine Sprache heißt **regulär** (bzw. vom Typ 3), wenn sie von einer Typ 3-Grammatik erzeugt wird.
- Eine Grammatik  $G = (V, \Sigma, P, S)$  ist vom Typ 3 (bzw. regulär), wenn alle Produktionen von der Form

$$A \rightarrow a \text{ oder } A \rightarrow aB$$

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

## Deterministische endliche Automaten

### Die informelle Kurzfassung:




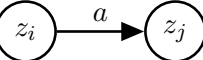
- Endliche Automaten lesen Zeichenweise ein Eingabewort
- Wechseln dabei den Zustand (eindeutig)
- Nur endlich viele Zustände
- Starten im Startzustand
- Nach Lesen der Eingabe: Akzeptieren oder Verwerfen
- Akzeptieren = in einem Endzustand
- Verwerfen = in keinem Endzustand
- Akzeptierte Sprache = alle Worte, für die der Automat akzeptiert

**Definition (Deterministischer Endlicher Automat, DFA)**

Ein **deterministischer endlicher Automat** (deterministic finite automaton, DFA) ist ein 5-Tupel  $M = (Z, \Sigma, \delta, z_0, E)$  wobei

- $Z$  ist eine endliche Menge von **Zuständen**,
- $\Sigma$  ist das (endliche) **Eingabealphabet** mit  $(Z \cap \Sigma) = \emptyset$ ,
- $z_0 \in Z$  ist der **Startzustand**,
- $E \subseteq Z$  ist die Menge der **Endzustände** (oder auch **akzeptierende Zustände**) und
- $\delta : Z \times \Sigma \rightarrow Z$  ist die **Zustandsüberföhrungsfunktion** (oder nur **Überföhrungsfunktion**).

Für DFA  $M = (Z, \Sigma, \delta, z_0, E)$

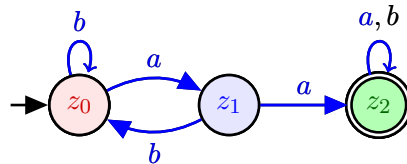
- Für  $z \in Z$  gibt es Knoten 
- Startzustand  $z_0 \in Z$ : eingehender Pfeil 
- Endzustände  $z \in E$ : doppelte Kreise 
- Übergänge  $\delta(z_i, a) = z_j$  als Kante 



Beispiel

DFA  $M = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_2\})$  mit

- $\delta(z_0, a) = z_1$      $\delta(z_1, b) = z_0$
- $\delta(z_0, b) = z_0$      $\delta(z_2, a) = z_2$
- $\delta(z_1, a) = z_2$      $\delta(z_2, b) = z_2$



Abarbeitung von **abbaaa**:

- Starte in  $z_0$
- Lese  $a$  und wechsele in  $z_1$
- Lese  $b$  und wechsele in  $z_0$
- Lese  $b$  und wechsele in  $z_0$
- Lese  $a$  und wechsele in  $z_1$
- Lese  $a$  und wechsele in  $z_2$
- Lese  $a$  und wechsele in  $z_2$
- Akzeptiere

Abarbeitung von **bbab**:

- Starte in  $z_0$
- Lese  $b$  und wechsele in  $z_0$
- Lese  $b$  und wechsele in  $z_0$
- Lese  $a$  und wechsele in  $z_1$
- Lese  $b$  und wechsele in  $z_0$
- **Verwerfe**

Akzeptierte Sprachen eines DFA

**Definition (Akzeptierte Sprache eines DFA)**

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

Wir definieren die Funktion  $\hat{\delta} : Z \times \Sigma^* \rightarrow Z$  durch

$$\hat{\delta}(z, \varepsilon) := z \text{ und } \hat{\delta}(z, aw) := \hat{\delta}(\delta(z, a), w)$$

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

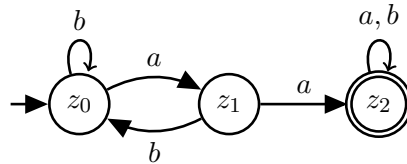
$$L(M) := \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E\}.$$

*$\hat{\delta}$  wendet  $\delta$  solange an, bis das Eingabewort abgearbeitet ist*

## Beispiel

DFA  $M = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_2\})$  mit

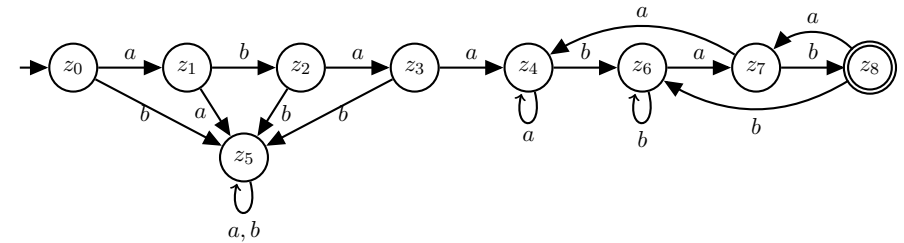
$$\begin{aligned} \delta(z_0, a) &= z_1 & \delta(z_1, b) &= z_0 \\ \delta(z_0, b) &= z_0 & \delta(z_2, a) &= z_2 \\ \delta(z_1, a) &= z_2 & \delta(z_2, b) &= z_2 \end{aligned}$$



$$L(M) = \{uaav \mid uv \in \{a, b\}^*\}$$

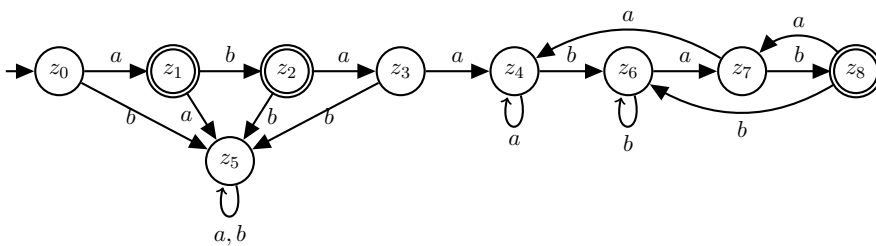
## Beispiel: DFA konstruieren

Konstruiere DFA über  $\Sigma = \{a, b\}$  der alle Worte akzeptiert, die mit  $abaa$  beginnen und mit  $bab$  enden:



## Beispiel: DFA konstruieren (2)

Zusätzlich die Wörter  $a$  und  $ab$  akzeptieren



## Lauf

### Definition

Sei  $M = (Z, \Sigma, \delta, z_0, E)$  ein DFA und  $w \in \Sigma^*$  mit  $|w| = n$ . Die Folge von Zuständen  $q_0, \dots, q_n$  mit  $q_0 = z_0$  und  $q_i = \delta(q_{i-1}, w[i])$  bezeichnet man als **Lauf** von  $M$  für Wort  $w$ .

Für eine solchen Lauf schreiben wir auch:

$$q_0 \xrightarrow{w[1]} q_1 \xrightarrow{w[2]} \dots \xrightarrow{w[n-1]} q_{n-1} \xrightarrow{w[n]} q_n$$

## DFAs akzeptieren reguläre Sprachen

### Theorem 4.2.1

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

Beweis: Sei  $G = (V, \Sigma, P, S)$  die reguläre Grammatik mit

$$V = Z, \quad S = z_0, \quad P = \begin{aligned} &\{z_i \rightarrow az_j \mid \delta(z_i, a) = z_j\} \\ &\cup \{z_i \rightarrow a \mid \delta(z_i, a) = z_j \wedge z_j \in E\} \\ &\cup \{z_0 \rightarrow \varepsilon \mid \text{falls } z_0 \in E\} \end{aligned}$$

Offensichtlich  $\varepsilon \in L(M) \iff \varepsilon \in L(G)$ . Ansonsten:

$$w = a_1 \cdots a_m \in L(M)$$

g.d.w. es gibt  $z_1, \dots, z_m \in Z$  mit  $\delta(z_{i-1}, a_i) = z_i$  und  $z_m \in E$ .

g.d.w.  $z_0 \Rightarrow_G a_1 z_1$ , für  $1 \leq i < m$ :  $a_1 \cdots a_{i-1} z_{i-1} \Rightarrow_G a_1 \cdots a_i z_i$  und

$$a_1 \cdots a_{m-1} z_{m-1} \Rightarrow_G a_1 \cdots a_m, \text{ d.h. } z_0 \Rightarrow_G^* a_1 \cdots a_m$$

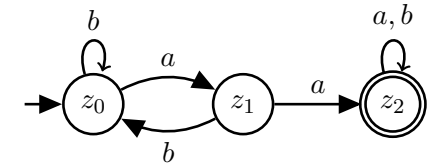
g.d.w.  $w = a_1 \cdots a_m \in L(G)$

Daher gilt  $L(M) = L(G)$  und somit ist  $L(M)$  regulär.  $\square$

## Beispiel: Konstruktion Typ 3-Grammatik aus DFA

DFA  $M = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_2\})$  mit

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



Die erzeugte reguläre Grammatik dazu ist:

$G = (\{z_0, z_1, z_2\}, \{a, b\}, P, z_0)$  mit

$$P = \begin{aligned} &\{z_0 \rightarrow az_1 \mid bz_0, \\ &\quad z_1 \rightarrow az_2 \mid a \mid bz_0, \\ &\quad z_2 \rightarrow az_2 \mid a \mid bz_2 \mid b\} \end{aligned}$$

## Wird jede reguläre Sprache durch einen DFA akzeptiert?

- Der vorherige Beweis konstruiert:  
„für jeden DFA gibt es eine äquivalente reguläre Grammatik“
- Für die andere Richtung wäre notwendig  
„für jede reguläre Grammatik gibt es einen äquivalenten DFA“

Problem:

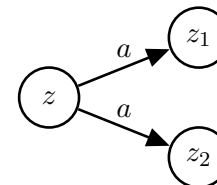
- Produktionen:  $A \rightarrow aA_1$  und  $A \rightarrow aA_2$  können in Grammatiken vorkommen
- Konstruktion des deterministischen Automaten unklar.

Daher: Beweis, dass DFAs alle regulären Sprachen akzeptieren, erfolgt auf Umwegen

## Nichtdeterministische Endliche Automaten

Ideen

- Zustandswechsel nicht eindeutig, sondern nichtdeterministisch in einen von mehreren möglichen
- D.h. der Automat darf sozusagen „raten“ welchen Nachfolgezustand er wählt
- Im Zustandsgraph erlaubt:



- Technisch:
  - DFA  $\delta : Z \times \Sigma \rightarrow Z$  und ein Startzustand
  - NFA  $\delta : Z \times \Sigma \rightarrow \mathcal{P}(Z)$  und Menge von Startzuständen

### Definition

Ein **nichtdeterministischer endlicher Automat** (nondeterministic finite automaton, NFA) ist ein 5-Tupel  $(Z, \Sigma, \delta, S, E)$  wobei

- $Z$  ist eine endliche Menge von **Zuständen**,
- $\Sigma$  ist das (endliche) **Eingabealphabet** mit  $(Z \cap \Sigma) = \emptyset$ ,
- $S \subseteq Z$  ist die Menge der **Startzustände**,
- $E \subseteq Z$  ist die Menge der **Endzustände** und
- $\delta : Z \times \Sigma \rightarrow \mathcal{P}(Z)$  ist die **Zustandsüberföhrungsfunktion**

## Beispiel: Leere Menge von Startzuständen

Sei  $M = (Z, \Sigma, \delta, \emptyset, E)$  ein NFA.  
Dann ist  $L(M) = \emptyset$ .

„Ein Wort  $w$  wird vom NFA akzeptiert, wenn es einen Pfad von einem Startzustand zum Endzustand entlang  $w$  gibt“

### Definition (Akzeptierte Sprache eines NFA)

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

Wir definieren  $\hat{\delta} : (\mathcal{P}(Z) \times \Sigma^*) \rightarrow \mathcal{P}(Z)$  induktiv durch:

$$\begin{aligned}\hat{\delta}(X, \varepsilon) &:= X \text{ f\u00fcr alle } X \subseteq Z \\ \hat{\delta}(X, aw) &:= \bigcup_{z \in X} \hat{\delta}(\delta(z, a), w) \text{ f\u00fcr alle } X \subseteq Z\end{aligned}$$

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

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(S, w) \cap E \neq \emptyset\}$$

## Lauf beim NFA

### Definition

Sei  $M = (Z, \Sigma, \delta, S, E)$  ein NFA und  $w \in \Sigma^*$  mit  $|w| = n$ .

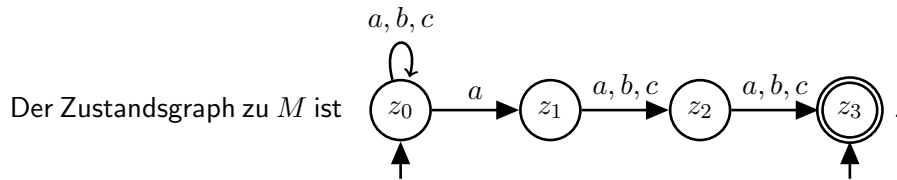
Eine Folge von Zuständen  $q_0, \dots, q_n$  mit  $q_0 \in S$  und  $q_i \in \delta(q_{i-1}, w[i])$  bezeichnet man als **Lauf** von  $M$  f\u00fcr Wort  $w$ .

Beachte: W\u00e4hrend es bei DFA's genau einen Lauf pro Wort gibt, kann es bei NFA's mehrere geben.

## Beispiel

Sei  $M = (\{z_0, z_1, z_2, z_3\}, \{a, b, c\}, \delta, \{z_0, z_3\}, \{z_3\})$  ein NFA mit

$$\begin{array}{llll} \delta(z_0, a) = \{z_0, z_1\} & \delta(z_1, a) = \{z_2\} & \delta(z_2, a) = \{z_3\} & \delta(z_3, a) = \emptyset \\ \delta(z_0, b) = \{z_0\} & \delta(z_1, b) = \{z_2\} & \delta(z_2, b) = \{z_3\} & \delta(z_3, b) = \emptyset \\ \delta(z_0, c) = \{z_0\} & \delta(z_1, c) = \{z_2\} & \delta(z_2, c) = \{z_3\} & \delta(z_3, c) = \emptyset \end{array}$$



$$L(M) = \{\varepsilon\} \cup \{uaw \mid u \in \{a, b, c\}^*, w \in \{a, b, c\}^2\}$$

## Jede reguläre Sprache wird durch NFA erkannt

### Theorem 4.4.1

Für jede reguläre Sprache  $L$  gibt es einen NFA  $M$  mit  $L(M) = L$ .

Beweis:

- Sei  $G = (V, \Sigma, P, S)$  eine reguläre Grammatik mit  $L(G) = L$ .
- Sei  $M = (Z, \Sigma, \delta, S', E)$  ein NFA mit  $Z = V \cup \{z_E\}$  ( $z_E$  neu),  $S' = \{S\}$  und  $E = \{z_E, S\}$  falls  $S \rightarrow \varepsilon \in P$ , sonst  $E = \{z_E\}$ , und
 
$$\delta(A, a) := \{B \mid A \rightarrow aB \in P\} \cup \{z_E \mid \text{falls } A \rightarrow a \in P\}$$

$$\delta(z_E, a) := \emptyset \text{ für alle } a \in \Sigma.$$
- Es gilt:  $\varepsilon \in L(M) \iff \varepsilon \in L(G)$ .
- Für  $w = a_1 \cdots a_n$  gilt:
 
$$w \in L(G) \text{ g.d.w. } S \Rightarrow_G a_1 A_1 \Rightarrow_G \dots \Rightarrow_G a_1 \cdots a_{n-1} A_{n-1} \Rightarrow_G a_1 \cdots a_n$$
 g.d.w. Es gibt Zustände  $A_1, \dots, A_{n-1}$  mit  $A_1 \in \delta(S, a_1)$ ,  $A_{i+1} \in \delta(A_i, a_{i+1})$  für  $1 \leq i \leq n-2$  und  $z_E \in \delta(A_{n-1}, a_n)$ 
 g.d.w.  $w \in L(M)$

## Beispiel: Konstruktion NFA aus Typ 3-Grammatik

Betrachte die reguläre Grammatik  $G = (V, \Sigma, P, A)$  mit  $V = \{A, B, C, D\}$ ,  $\Sigma = \{a, b, c\}$  und

$$\begin{array}{l} P = \{ A \rightarrow \varepsilon \mid aB \mid bB \mid cB \mid aC, \\ \quad B \rightarrow aB \mid bB \mid cB \mid aC, \\ \quad C \rightarrow aD \mid bD \mid cD, \\ \quad D \rightarrow a \mid b \mid c \} \end{array}$$

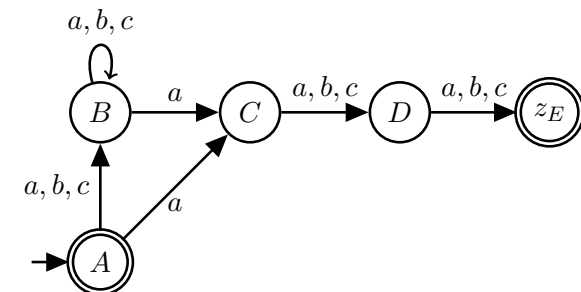
Konstruktion des dazu passenden NFA:  $M = (Z, \Sigma, \delta, S, E)$  mit

- $Z = V \cup \{z_E\} = \{A, B, C, D, z_E\}$ ,
- $E = \{A, z_E\}$ ,  $S = \{A\}$  und

$$\begin{array}{lllll} \delta(A, a) = \{B, C\} & \delta(B, a) = \{B, C\} & \delta(C, a) = \{D\} & \delta(D, a) = \{z_E\} & \delta(z_E, a) = \emptyset \\ \delta(A, b) = \{B\} & \delta(B, b) = \{B\} & \delta(C, b) = \{D\} & \delta(D, b) = \{z_E\} & \delta(z_E, b) = \emptyset \\ \delta(A, c) = \{B\} & \delta(B, c) = \{B\} & \delta(C, c) = \{D\} & \delta(D, c) = \{z_E\} & \delta(z_E, c) = \emptyset \end{array}$$

## Beispiel: Konstruktion NFA aus Typ 3-Grammatik (2)

Der Zustandsgraph zu  $M$  ist

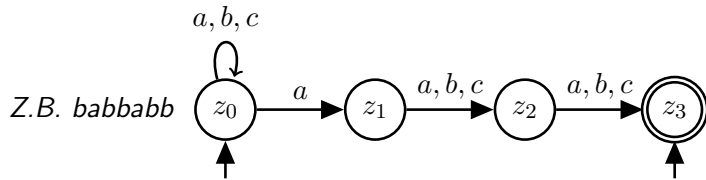


## Theorem 4.5.1 (Rabin & Scott 1959)

Jede von einem NFA akzeptierte Sprache ist auch durch einen DFA akzeptierbar.

Beweisidee:

- Konstruiere für einen gegebenen NFA einen DFA, sodass sich „der DFA **alle** Zustände merkt, in denen der NFA sein könnte“



- Konstruktion: Jede Teilmenge von Zuständen des NFA wird zu einem Zustand des DFA (daher: **Potenzmengenkonstruktion**)

Für NFA  $M = (Z, \Sigma, \delta, S, E)$  konstruieren wir den DFA  $M' = (Z', \Sigma, \delta', S', E')$  mit

- $Z' = \mathcal{P}(Z)$   
„Zustandsmenge ist Potenzmenge von  $Z$ “
- $S' = S$   
„Startzustand ist Menge  $S$  aller Startzustände von  $M$ “
- $E' = \{X \in Z' \mid (E \cap X) \neq \emptyset\}$   
„Jede Menge, die mind. einen Endzustand von  $E$  enthält, ist Endzustand in  $M'$ “
- $\delta'(X, a) = \bigcup_{z \in X} \delta(z, a) = \widehat{\delta}(X, a)$   
„ $\delta'(X, a)$  berechnet alle von einem Zustand in  $X$  aus über  $a$  erreichbaren Zustände.“

## Korrektheit der Potenzmengenkonstruktion

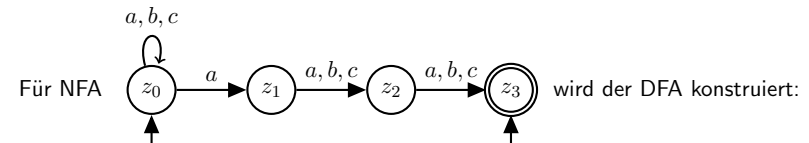
Wir beweisen, dass  $L(M) = L(M')$  gilt, indem wir zeigen:

$$w \in L(M) \text{ g.d.w. } w \in L(M')$$

- Fall  $w = \varepsilon$ :  
 $\varepsilon \in L(M)$  g.d.w.  $S \cap E \neq \emptyset$  g.d.w.  $S \in E'$  g.d.w.  $\varepsilon \in L(M')$

- Fall  $w = a_1 \cdots a_n \in \Sigma^*$ :  
 $w \in L(M)$   
 g.d.w.  $\widehat{\delta}(S, w) \cap E \neq \emptyset$   
 g.d.w. Es gibt Teilmengen  $Z_1, \dots, Z_n$  von  $Z$  mit  
 $\delta(S, a_1) = Z_1, \delta(Z_i, a_{i+1}) = Z_{i+1}$  für  $i = 1, \dots, n-1$   
 und  $Z_n \cap E \neq \emptyset$   
 g.d.w.  $\widehat{\delta}(S', w) \in E'$   
 g.d.w.  $w \in L(M')$

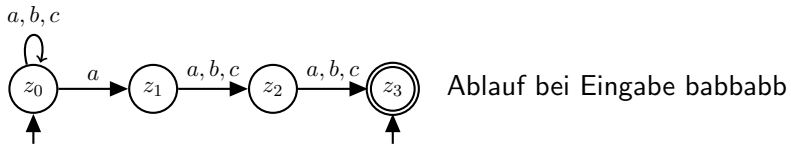
## Beispiel: Potenzmengenkonstruktion



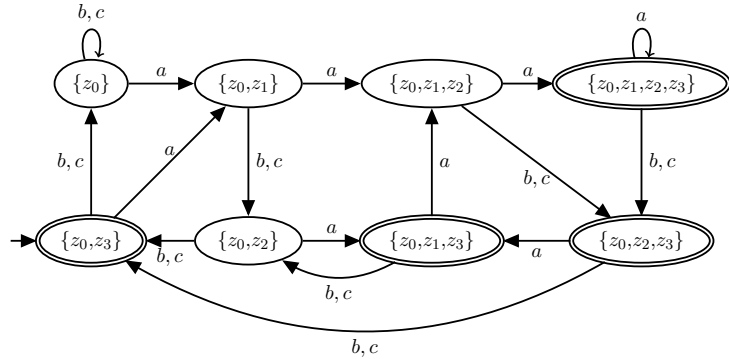
$M' = (\mathcal{P}(\{z_0, z_1, z_2, z_3\}), \{a, b, c\}, \delta', S', E')$  mit  
 $S' = \{z_0, z_3\}$   
 $E' = \{\{z_3\}, \{z_0, z_3\}, \{z_1, z_3\}, \{z_2, z_3\}, \{z_0, z_1, z_3\}, \{z_0, z_2, z_3\}, \{z_1, z_2, z_3\}, \{z_0, z_1, z_2, z_3\}\}$

$\delta'(\emptyset, d) = \emptyset$ für $d \in \{a, b, c\}$	$\delta'(\{z_1, z_3\}, d) = \{z_2\}$ für $d \in \{a, b, c\}$
$\delta'(\{z_0\}, a) = \{z_0, z_1\}$	$\delta'(\{z_2, z_3\}, d) = \{z_3\}$ für $d \in \{a, b, c\}$
$\delta'(\{z_0\}, d) = \{z_0\}$ für $d \in \{b, c\}$	$\delta'(\{z_0, z_1, z_2\}, a) = \{z_0, z_1, z_2, z_3\}$
$\delta'(\{z_1\}, d) = \{z_2\}$ für $d \in \{a, b, c\}$	$\delta'(\{z_0, z_1, z_2\}, d) = \{z_0, z_2, z_3\}$ für $d \in \{b, c\}$
$\delta'(\{z_2\}, d) = \{z_3\}$ für $d \in \{a, b, c\}$	$\delta'(\{z_0, z_1, z_3\}, a) = \{z_0, z_1, z_2\}$
$\delta'(\{z_3\}, d) = \emptyset$ für $d \in \{a, b, c\}$	$\delta'(\{z_0, z_1, z_3\}, d) = \{z_0, z_2\}$ für $d \in \{b, c\}$
$\delta'(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$	$\delta'(\{z_0, z_2, z_3\}, a) = \{z_0, z_1, z_3\}$
$\delta'(\{z_0, z_1\}, d) = \{z_0, z_2\}$ für $d \in \{b, c\}$	$\delta'(\{z_0, z_2, z_3\}, d) = \{z_0, z_3\}$ für $d \in \{b, c\}$
$\delta'(\{z_0, z_2\}, a) = \{z_0, z_1, z_3\}$	$\delta'(\{z_1, z_2, z_3\}, d) = \{z_2, z_3\}$ für $d \in \{a, b, c\}$
$\delta'(\{z_0, z_2\}, d) = \{z_0, z_3\}$ für $d \in \{b, c\}$	$\delta'(\{z_0, z_1, z_2, z_3\}, a) = \{z_0, z_1, z_2, z_3\}$
$\delta'(\{z_0, z_3\}, a) = \{z_0, z_1\}$	$\delta'(\{z_0, z_1, z_2, z_3\}, b) = \{z_0, z_2, z_3\}$
$\delta'(\{z_0, z_3\}, d) = \{z_0\}$ für $d \in \{b, c\}$	$\delta'(\{z_0, z_1, z_2, z_3\}, c) = \{z_0, z_2, z_3\}$
$\delta'(\{z_1, z_2\}, d) = \{z_2, z_3\}$ für $d \in \{a, b, c\}$	

## Beispiel: Potenzmengenkonstruktion (2)



DFA als Zustandsgraph (nur erreichbare Zustände):



## DFAs & NFAs sind Formalismen für Typ 3-Sprachen

### Theorem 4.5.4

DFAs und NFAs erkennen genau die regulären Sprachen.

Das folgt aus:

- Theorem 4.2.1: Sei  $M$  ein DFA. Dann ist  $L(M)$  regulär.
- Theorem 4.4.1: Für jede reguläre Sprache  $L$  gibt es einen NFA  $M$  mit  $L(M) = L$ .
- Theorem 4.5.1: Jede von einem NFA akzeptierte Sprache ist auch durch einen DFA akzeptierbar.
- Jeder DFA kann leicht auch als NFA interpretiert werden

## Größe des DFAs vs NFAs (1)

- Sei  $M$  ein NFA mit  $n$  Zuständen.
- Der durch die Potenzmengenkonstruktion erstellte DFA hat  $2^n$  Zustände!
- D.h. der Platz explodiert uns!
- Frage: Geht es besser (unsere Kodierung ist zu einfach) oder nicht?
- Das folgende Lemma zeigt, dass es nicht wirklich besser geht

## Größe des DFAs vs NFAs (2)

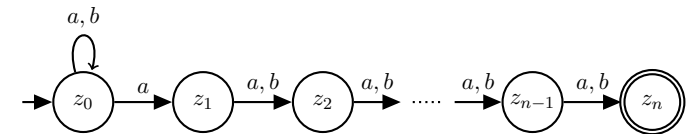
### Lemma

Sei  $L_n = \{uav \mid u \in \{a, b\}^*, v \in \{a, b\}^{n-1}\}$  für  $n \in \mathbb{N}_{>0}$ .

(Sprache aller Wörter aus  $\{a, b\}^*$ , die an  $n$ -letzter Stelle ein  $a$  haben).

- Es gibt NFA  $M_n$  mit  $L(M_n) = L_n$  und  $M_n$  hat  $n + 1$  Zustände.
- Jeder DFA  $M'_n$  mit  $L(M'_n) = L_n$ , hat mindestens  $2^n$  Zustände.

Beweis (Teil 1): Sei  $M_n$  der folgende NFA:



$L(M_n) = L_n$ , denn:

- zum Akzeptieren müssen  $z_0, z_1, \dots, z_n$  nacheinander durchlaufen werden, was genau mit Wörtern  $av$  mit  $v \in \{a, b\}^{n-1}$  möglich ist
- In  $z_0$  kann zuvor jedes  $u \in \{a, b\}^*$  gelesen werden (Verbleib in  $z_0$ ).



## Größe des DFAs vs NFAs (3)

Beweis (Teil 2): Beweis durch Widerspruch.

- Annahme: Es gibt  $n \in \mathbb{N}_{>0}$  und DFA  $M' = (Z, \{a, b\}, \delta, z_0, E)$  mit  $L(M') = L_n = \{uav \mid u \in \{a, b\}^*, v \in \{a, b\}^{n-1}\}$  und  $|Z| < 2^n$
- Menge  $W = \{a, b\}^n$  enthält  $2^n$  Worte der Länge  $n$  und da  $|Z| < 2^n$ , muss es  $w \neq w' \in W$  geben mit  $\hat{\delta}(z_0, w) = \hat{\delta}(z_0, w') = z_i$
- Sei  $j$  die erste Position, an der sich  $w$  und  $w'$  unterscheiden.

Falls  $j = 1$ , dann ist o.B.d.A.  $w = au \in L_n$  aber  $w' = bu' \notin L_n$  und  $z_i \in E$  und  $z_i \notin E$  müsste gleichzeitig gelten. **Widerspruch!**

Falls  $j > 1$ : O.B.d.A.  $w = uav$  und  $w' = ubv'$  mit  $|v| = |v'| = n - j$

$$\begin{aligned} \text{Sei } w_0 &= wb^{j-1} = uavb^{j-1} \\ w'_0 &= w'b^{j-1} = ubv'b^{j-1} \end{aligned}$$

Dann muss gelten  $\hat{\delta}(w_0) = \hat{\delta}(w'_0)$ , da  $\hat{\delta}(uav) = z_i = \hat{\delta}(ubv')$ .

Aber  $w_0 \in L_n$  und  $w'_0 \notin L_n$ , **Widerspruch!** □

## NFAs mit ε-Übergängen

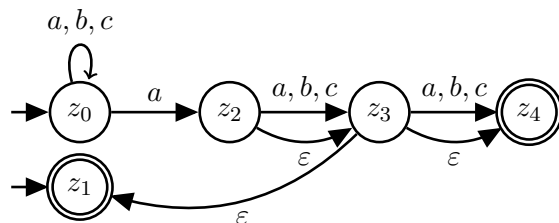
- ε-Übergänge erlauben Zustandswechsel **ohne** Lesen eines Zeichens (es wird sozusagen das leere Wort ε gelesen)
- Ausdruckskraft ändert sich mit ε-Übergängen nicht
- ε-Übergänge machen manche Konstruktionen einfacher.

### Definition (NFA mit ε-Übergängen)

Ein **nichtdeterministischer endlicher Automat mit ε-Übergängen** (NFA mit ε-Übergängen) ist ein Tupel  $M = (Z, \Sigma, \delta, S, E)$  wobei

- $Z$  ist eine endliche Menge von Zuständen,
- $\Sigma$  ist das (endliche) Eingabealphabet mit  $(Z \cap \Sigma) = \emptyset$ ,
- $S \subseteq Z$  ist die Menge der Startzustände,
- $E \subseteq Z$  ist die Menge der Endzustände und
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Z)$  ist die Zustandsüberföhrungsfunktion

## Beispiel: NFA mit ε-Übergängen



Akzeptierte Sprache:

alle Worte aus  $\{a, b, c\}^*$ , die an letzter, vorletzter, oder drittletzter Position ein  $a$  haben, und das leere Wort

## ε-Hülle

### Definition (ε-Hülle)

Sei  $M = (Z, \Sigma, \delta, S, E)$  ein NFA mit ε-Übergängen. Die ε-Hülle  $clos_\varepsilon(z)$  eines Zustands  $z \in Z$  ist induktiv definiert als die kleinste Menge von Zuständen, welche die folgenden Eigenschaften erfüllt:

- 1  $z \in clos_\varepsilon(z)$ .
- 2 Wenn  $z' \in clos_\varepsilon(z)$  und  $z'' \in \delta(z', \varepsilon)$ , dann ist auch  $z'' \in clos_\varepsilon(z)$ .

Für eine Zustandsmenge  $X \subseteq Z$  definieren wir

$$clos_\varepsilon(X) := \bigcup_{z \in X} clos_\varepsilon(z).$$

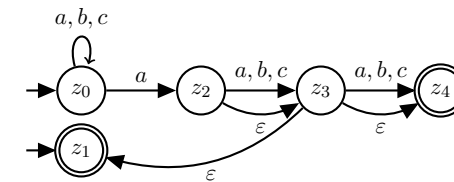
Die ε-Hülle fügt für eine Zustandsmenge alle durch ε-Übergänge erreichbaren Zustände hinzu.

## $\varepsilon$ -Hülle (2)

Die  $\varepsilon$ -Hülle für eine Zustandsmenge  $X \subseteq Z$  kann auch berechnet werden durch:

$$\text{clos}_\varepsilon(X) := \begin{cases} X, & \text{wenn } \bigcup_{z \in X} \delta(z, \varepsilon) \subseteq X \\ \text{clos}_\varepsilon(X \cup \bigcup_{z \in X} \delta(z, \varepsilon)), & \text{sonst} \end{cases}$$

## Beispiel



$$\begin{aligned} \text{clos}_\varepsilon(z_0) &= \{z_0\} \\ \text{clos}_\varepsilon(z_1) &= \{z_1\} \\ \text{clos}_\varepsilon(z_4) &= \{z_4\} \\ \text{clos}_\varepsilon(z_3) &= \{z_1, z_3, z_4\} \\ \text{clos}_\varepsilon(z_2) &= \{z_1, z_2, z_3, z_4\} \end{aligned}$$

## NFA mit $\varepsilon$ -Übergängen: Akzeptierte Sprache

### Akzeptierte Sprache eines NFA mit $\varepsilon$ -Übergängen

Sei  $M = (Z, \Sigma, \delta, S, E)$  ein NFA mit  $\varepsilon$ -Übergängen.  
Wir definieren  $\tilde{\delta} : (\mathcal{P}(Z) \times \Sigma^*) \rightarrow \mathcal{P}(Z)$  induktiv durch:

$$\begin{aligned} \tilde{\delta}(X, \varepsilon) &:= X \\ \tilde{\delta}(X, aw) &:= \bigcup_{z \in X} \tilde{\delta}(\text{clos}_\varepsilon(\delta(z, a)), w) \text{ für alle } X \subseteq Z \end{aligned}$$

Die von  $M$  akzeptierte Sprache ist

$$L(M) := \{w \in \Sigma^* \mid \tilde{\delta}(\text{clos}_\varepsilon(S), w) \cap E \neq \emptyset\}$$

## $\varepsilon$ -Übergänge ändern die Ausdruckskraft nicht (1)

### Satz 4.6.7

NFAs mit  $\varepsilon$ -Übergängen akzeptieren genau die regulären Sprachen.

Beweis „ $\Leftarrow$ “:

- Jede reguläre Sprache wird von einem „normalen“ NFA akzeptiert.
- Transformiere diesen NFA in einen NFA mit  $\varepsilon$ -Übergängen:  
Setze  $\delta(z, \varepsilon) = \emptyset$  für alle Zustände  $z$   
Offensichtlich ist die akzeptierte Sprache dieselbe.
- Daher wird jede reguläre Sprache von einem NFA mit  $\varepsilon$ -Übergängen akzeptiert.

## $\varepsilon$ -Übergänge ändern die Ausdruckskraft nicht (2)

Beweis „ $\Rightarrow$ “: Sei  $M = (Z, \Sigma, \delta, S, E)$  ein NFA mit  $\varepsilon$ -Übergängen.

- Konstruiere NFA  $M'$  mit  $L(M) = L(M')$ . Dann ist  $L(M)$  regulär.
- $M' = (Z, \Sigma, \delta', S', E)$  mit  $S' = \text{clos}_\varepsilon(S)$ ,  $\delta'(z, a) = \text{clos}_\varepsilon(\delta(z, a))$ .

$L(M) = L(M')$ :

- Wir zeigen  $\tilde{\delta}(\text{clos}_\varepsilon(X), w) = \hat{\delta}'(\text{clos}_\varepsilon(X), w)$  für alle  $X \subseteq Z$  und  $w \in \Sigma^*$ . Wir verwenden Induktion über die Wortlänge  $|w|$ .
- Basis:  $w = \varepsilon$ . Dann gilt  $\tilde{\delta}(\text{clos}_\varepsilon(X), \varepsilon) = \text{clos}_\varepsilon(X) = \hat{\delta}'(\text{clos}_\varepsilon(X), \varepsilon)$
- Schritt: Sei  $w = au$  mit  $a \in \Sigma$ . Wir formen um:

$$\tilde{\delta}(\text{clos}_\varepsilon(X), au) \stackrel{\text{Def. } \tilde{\delta}}{=} \tilde{\delta} \bigcup_{z \in \text{clos}_\varepsilon(X)} \tilde{\delta}(\text{clos}_\varepsilon(\delta(z, a)), u) \stackrel{\text{I.H.}}{=} \bigcup_{z \in \text{clos}_\varepsilon(X)} \hat{\delta}'(\text{clos}_\varepsilon(\delta(z, a)), u)$$

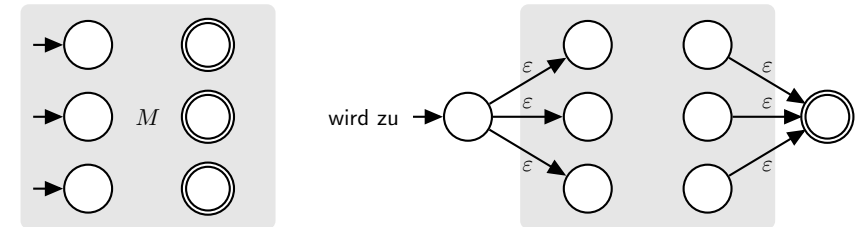
$$\stackrel{\text{Def. } \delta'}{=} \bigcup_{z \in \text{clos}_\varepsilon(X)} \hat{\delta}'(\delta'(z, a), u) \stackrel{\text{Def. } \hat{\delta}'}{=} \hat{\delta}'(\text{clos}_\varepsilon(X), au)$$

## Eindeutige Start- und Endzustände

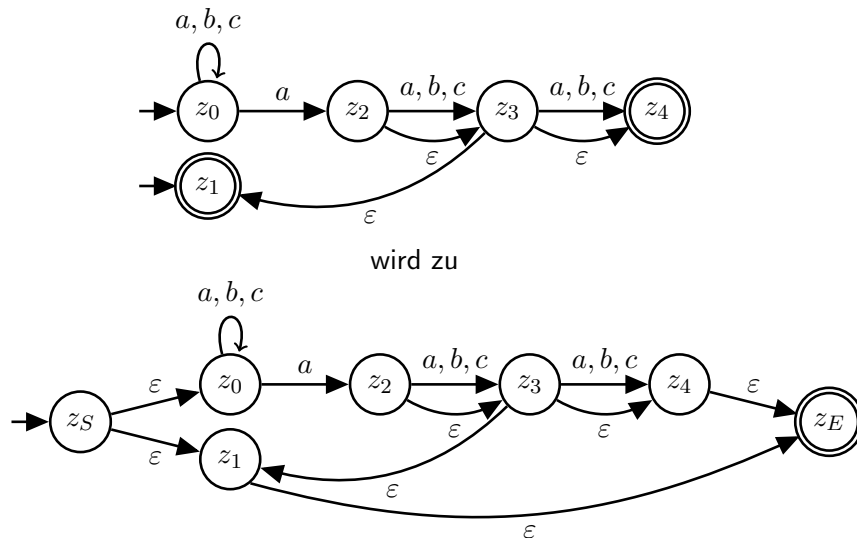
### Satz 4.6.8

Für jeden NFA  $M$  mit  $\varepsilon$ -Übergängen gibt es einen NFA  $M'$  mit  $\varepsilon$ -Übergängen, sodass  $L(M) = L(M')$  und  $M'$  genau einen Startzustand und genau einen Endzustand hat, wobei diese beiden Zustände verschieden sind.

Beweis: Konstruiere  $M'$  aus  $M$ , durch Hinzufügen eines neuen Start- und eines neuen Endzustand mit  $\varepsilon$ -Übergängen:



## Beispiel



## Reguläre Ausdrücke

- Reguläre Ausdrücke sind (wie Automaten und Grammatiken) ein Formalismus zur Repräsentation von Sprachen.
- Praktische Verwendung: Regex-Bibliotheken in Programmiersprachen zum Suchen und Ersetzen von Zeichenketten (verwenden meist erweiterte reguläre Ausdrücke)
- Aufbau regulärer Ausdrücke: Basisausdrücke und Operatoren zum Zusammensetzen.

### Definition (Regulärer Ausdruck)

Sei  $\Sigma$  ein Alphabet. Ein **regulärer Ausdruck** über  $\Sigma$  ist induktiv definiert:

- $\emptyset$  ist ein regulärer Ausdruck
- $\varepsilon$  ist ein regulärer Ausdruck
- $a$  mit  $a \in \Sigma$  ist ein regulärer Ausdruck
- Wenn  $\alpha_1$  und  $\alpha_2$  reguläre Ausdrücke sind, dann auch  $\alpha_1\alpha_2$
- Wenn  $\alpha_1$  und  $\alpha_2$  reguläre Ausdrücke sind, dann auch  $(\alpha_1|\alpha_2)$
- Wenn  $\alpha$  regulärer Ausdruck ist, dann auch  $(\alpha)^*$

## Beispiele

- $(a|b)^*aa(a|b)^*$  erzeugt alle Worte über  $\{a, b\}$  die zwei aufeinanderfolgende  $a$ 's enthalten
- $(\varepsilon|((a|b|c)^*a(a|b|c)(a|b|c)(a|b|c)))$  erzeugt alle Worte über  $\{a, b, c\}$ , die an viertletzter Stelle ein  $a$  haben und das leere Wort
- $((0|1|2|3|4|5|6|7|8|9)|1(0|1|2|3|4|5|6|7|8|9)|(2(0|1|2|3)))$  :  $((0|1|2|3|4|5)(0|1|2|3|4|5|6|7|8|9))$  erzeugt alle Uhrzeiten im 24-Stunden-Format
- Eine endliche Sprache  $S = \{w_1, \dots, w_n\}$  wird durch  $(w_1|\dots|w_n)$  erzeugt.

### Erzeugte Sprache

Die von einem regulären Ausdruck  $\alpha$  **erzeugte Sprache**  $L(\alpha)$  ist induktiv über dessen Struktur definiert:

$$\begin{aligned}L(\emptyset) &:= \emptyset \\L(\varepsilon) &:= \{\varepsilon\} \\L(a) &:= \{a\} \quad \text{für } a \in \Sigma \\L(\alpha_1\alpha_2) &:= L(\alpha_1)L(\alpha_2) = \{uv \mid u \in L(\alpha_1), v \in L(\alpha_2)\} \\L(\alpha_1|\alpha_2) &:= L(\alpha_1) \cup L(\alpha_2) \\L((\alpha)^*) &:= L(\alpha)^*\end{aligned}$$

Für alle regulären Ausdrücke  $\alpha_1, \alpha_2, \alpha_3$  gilt:

$$L((\alpha_1|\alpha_2)|\alpha_3) = L(\alpha_1|(\alpha_2|\alpha_3))$$

Daher lassen wir Klammern weg und schreiben  $(\alpha_1|\alpha_2|\dots|\alpha_n)$ .

## Satz von Kleene

### Theorem 4.7.4 (Satz von Kleene)

Reguläre Ausdrücke erzeugen genau die regulären Sprachen.

Beweis in zwei Teilen:

- 1 Jede von einem regulären Ausdruck erzeugte Sprache ist regulär.
- 2 Für jede reguläre Sprache gibt es einen regulären Ausdruck, der sie erzeugt.

## Beweis: Satz von Kleene (1)

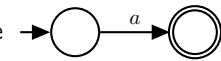


1. Jede von einem regulären Ausdruck erzeugte Sprache ist regulär.

Beweis:

- Wir konstruieren für regulären Ausdruck  $\alpha$  einen NFA  $M_\alpha$  mit  $\varepsilon$ -Übergängen und eindeutigen Start- und Endzuständen, sodass  $L(M_\alpha) = L(\alpha)$ .

- Induktion über die Struktur von  $\alpha$

- Basisfälle:

- Für  $\alpha = a \in \Sigma$  konstruiere 
- Für  $\alpha = \varepsilon$  konstruiere 
- Für  $\alpha = \emptyset$  konstruiere 

In allen Fällen ist  $L(\alpha) = L(M_\alpha)$  offensichtlich

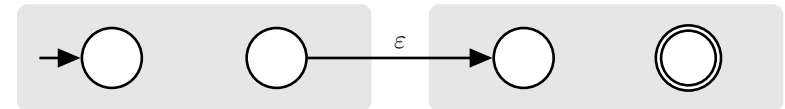
## Beweis: Satz von Kleene (2)

- Induktionsschritt: Betrachte den Aufbau von  $\alpha$  (3 Fälle)

- Für  $\alpha = \alpha_1\alpha_2$ , liefert die I.H.  $M_{\alpha_1}, M_{\alpha_2}$ .

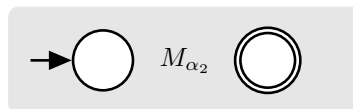


Konstruiere daraus  $M_\alpha$ :

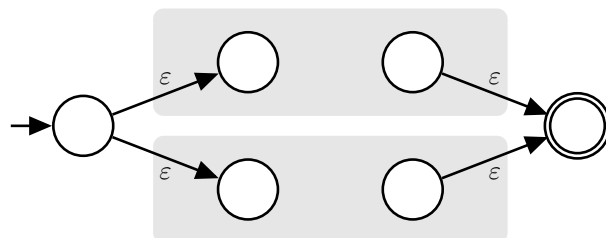


## Beweis: Satz von Kleene (3)

- Für  $\alpha = (\alpha_1|\alpha_2)$  liefert die I.H.  $M_{\alpha_1}, M_{\alpha_2}$ :

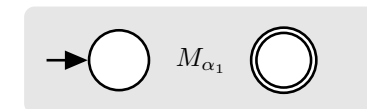


Konstruiere daraus  $M_\alpha$ :

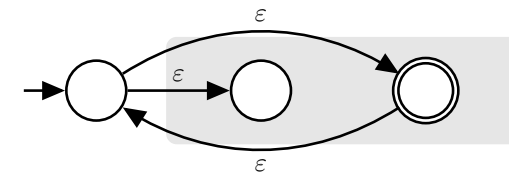


## Beweis: Satz von Kleene (4)

- Für  $\alpha = (\alpha_1)^*$  liefert die I.H.  $M_{\alpha_1}$



Konstruiere daraus  $M_\alpha$ :



## Beweis: Satz von Kleene (5)

2. Für jede reg. Sprache  $L$  gibt es einen regulären Ausdruck  $\alpha$  mit  $L(\alpha) = L$

Beweis:

- Sei DFA  $M = (\{z_1, \dots, z_n\}, \Sigma, \delta, z_1, E)$  mit  $L(M) = L$  gegeben.
- Für  $w \in \Sigma^*$  und  $z_i \in Z$  mit  $\widehat{\delta}(z_i, w) = z_j$  sei  $visit_i(w) = q_1, \dots, q_m$  die Folge der besuchten Zustände (wobei  $q_1 = z_i$  und  $q_m = z_j$ ).
- Wir definieren:

$$L_{i,j}^k = \left\{ w \in \Sigma^* \mid \begin{array}{l} \widehat{\delta}(z_i, w) = z_j \text{ und } visit_i(w) = q_1, \dots, q_m, \\ \text{sodass für } 1 < l < m: \text{ wenn } q_l = z_p \text{ dann } p \leq k \end{array} \right\}$$

$L_{i,j}^k$  enthält die Worte, die  $M$  von Zustand  $z_i$  zu Zustand  $z_j$  führen ohne dabei Zwischenzustände mit Index größer als  $k$  zu benutzen.

- Mit Induktion über  $k$  zeigen wir, dass es reguläre Ausdrücke  $\alpha_{i,j}^k$  mit  $L(\alpha_{i,j}^k) = L_{i,j}^k$  gibt.

## Beweis: Satz von Kleene (6)

Zur Erinnerung:  $L_{i,j}^k$  enthält die Worte, die  $M$  von Zustand  $z_i$  zu Zustand  $z_j$  führen ohne dabei Zwischenzustände mit Index größer als  $k$  zu benutzen.

Basis:  $k = 0$

- Wenn  $i \neq j$ , dann ist  $L_{i,j}^0 = \{a \in \Sigma \mid \delta(z_i, a) = z_j\}$ .  
Falls  $L_{i,j}^0 = \{a_1, \dots, a_q\}$ , dann gilt  $L(\alpha_{i,j}^0) = L_{i,j}^0$  für  $\alpha_{i,j}^0 = (a_1 | \dots | a_q)$ .  
Falls  $L_{i,j}^0 = \emptyset$ , dann gilt  $L(\alpha_{i,j}^0) = L_{i,j}^0$  mit  $\alpha_{i,j}^0 = \emptyset$ .
- Wenn  $i = j$ , dann ist  $L_{i,i}^0 = \{\varepsilon\} \cup \{a \in \Sigma \mid \delta(z_i, a) = z_i\}$ .  
Sei  $L_{i,i}^0 = \{\varepsilon, a_1, \dots, a_q\}$ .  
Dann gilt  $L(\alpha_{i,i}^0) = L_{i,i}^0$  für  $\alpha_{i,i}^0 = (\varepsilon | a_1 | \dots | a_q)$ .

## Beweis: Satz von Kleene (7)

Zur Erinnerung:  $L_{i,j}^k$  enthält die Worte, die  $M$  von Zustand  $z_i$  zu Zustand  $z_j$  führen ohne dabei Zwischenzustände mit Index größer als  $k$  zu benutzen.

Induktionsschritt:  $k \rightarrow k + 1$

$$L_{i,j}^{k+1} = L_{i,j}^k \cup L_{i,k+1}^k (L_{k+1,k+1}^k)^* L_{k+1,j}^k,$$

denn entweder läuft  $M$  ohne Zustand  $z_{k+1}$  zu besuchen, oder der Lauf kann in 3 Teile gespalten werden:

- 1 Lauf von  $z_i$  bis zum ersten Besuch des Zustands  $z_{k+1}$  (abgedeckt durch  $L_{i,k+1}^k$ )
- 2 Mehrmaliges, zyklisches Besuchen von  $k + 1$  (beliebig oft) (abgedeckt durch  $(L_{k+1,k+1}^k)^*$ )
- 3 Letztmaliges Verlassen von  $z_{k+1}$  und Lauf bis zu  $z_j$  (abgedeckt durch  $L_{k+1,j}^k$ )

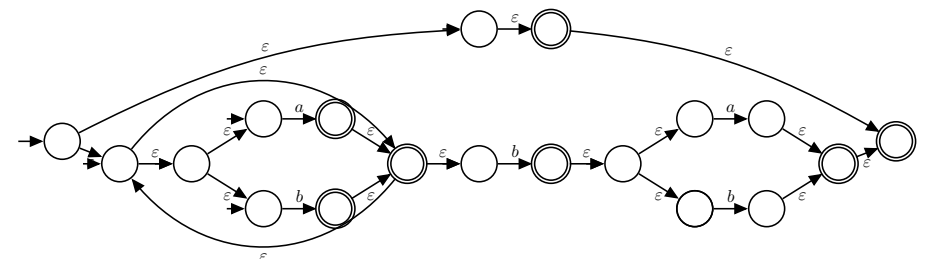
Daher gilt  $\alpha_{i,j}^{k+1} = (\alpha_{i,k+1}^k | \alpha_{k+1,k+1}^k (\alpha_{k+1,k+1}^k)^* | \alpha_{k+1,j}^k)$  und  $L(\alpha_{i,j}^{k+1}) = L_{i,j}^{k+1}$ .

## Beispiel: Regulärer Ausdruck $\rightarrow$ NFA mit $\varepsilon$ -Übergängen

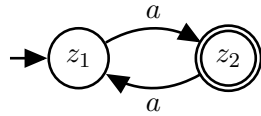
NFA zum regulären Ausdruck

$$(\varepsilon | (a|b)^* b (a|b))$$

konstruieren:



## Beispiel: DFA → regulärer Ausdruck



Regulärer Ausdruck dazu:

$$\begin{aligned} \alpha_{1,2}^2 &= (\alpha_{1,2}^1 | \alpha_{1,2}^1 (\alpha_{2,2}^1)^* \alpha_{2,2}^1) \\ &= ((a | \varepsilon(\varepsilon)^* a) | (a | \varepsilon(\varepsilon)^* a) (\varepsilon | a(\varepsilon)^* a)^* (\varepsilon | a(\varepsilon)^* a)) \\ &= (a | a(aa)^*) \text{ (durch Vereinfachung)} \end{aligned}$$

denn

$$\begin{aligned} \alpha_{1,1}^0 &= \varepsilon & \alpha_{2,2}^0 &= \varepsilon & \alpha_{1,2}^0 &= a & \alpha_{2,1}^0 &= a \\ \alpha_{1,2}^1 &= (\alpha_{1,2}^0 | \alpha_{1,1}^0 (\alpha_{1,1}^0)^* \alpha_{1,2}^0) = (a | \varepsilon(\varepsilon)^* a) \\ \alpha_{2,2}^1 &= (\alpha_{2,2}^0 | \alpha_{2,1}^0 (\alpha_{1,1}^0)^* \alpha_{2,2}^0) = (\varepsilon | a(\varepsilon)^* a) \end{aligned}$$

## Zusammenfassung: Formalismen für reguläre Sprachen

