



## 1.2 Reguläre Sprachen (Typ 3)

↪ (nicht)deterministische endliche Automaten

↪ reguläre Ausdrücke

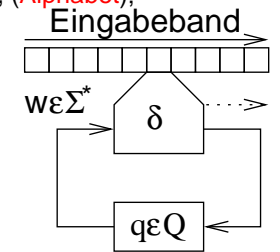
- Nichtreguläre Sprachen
- Abschlusseigenschaften



### 1.2.1 (Deterministische) endliche Automaten

Ein deterministischer endlicher Automat besteht aus:  
(auch endl. Akzeptor oder deterministic finite automaton=DFA)

- $Q$ , einer endlichen Menge von Zuständen;
- $\Sigma$ , einer endlichen Menge von Eingabesymbolen, (Alphabet);
- $\delta: Q \times \Sigma \rightarrow Q$ , einer Übergangsfunktion;
- $s \in Q$ , einem Startzustand;
- $F \subseteq Q$ , einer Menge von Endzuständen.



### Was tut ein endlicher Automat?

Wir erweitern die Def. von  $\delta$  für Wörter:

$$\hat{\delta}(q, \varepsilon) = q$$

$$\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w) \quad \text{— ein Zustandsübergang je}$$

Eingabezeichen

$A = (Q, \Sigma, \delta, s, F)$  akzeptiert die Sprache

$$L(A) := \{ w \in \Sigma^* : \hat{\delta}(s, w) \in F \} \quad \text{(Schöning: } T(A) \text{)}$$

### Äquivalente Definition:

$$\hat{\delta}(q, \varepsilon) = q$$

$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

Beweis: Übung



### Graphinterpretation

$$A = (Q, \Sigma, \delta, s, F)$$

$$G_A = (Q, E)$$

mit  $e = (q, q') \in E$ , Beschriftung  $\ell(e) = a$  falls  $q' = \delta(q, a)$

Multigraph!

#### Lemma:

$$\forall w \in \Sigma^* : w \in L(A) \Leftrightarrow$$

$$\exists \text{Abarbeitungspfad } P = sq_1q_2 \cdots f = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} f$$

mit  $f \in F$ , der mit  $w = a_1a_2 \cdots a_k$  beschriftet ist.

Beweis: Übung

#### Terminologie:

Wenn wir von Pfaden in  $A$  reden, meinen wir Pfade in  $G_A$ .



### Notation für Pfade

$P = sq_1q_2 \dots f$  Folge von **Knoten**

$P = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} f$  Folge von (direkten) **Übergängen**

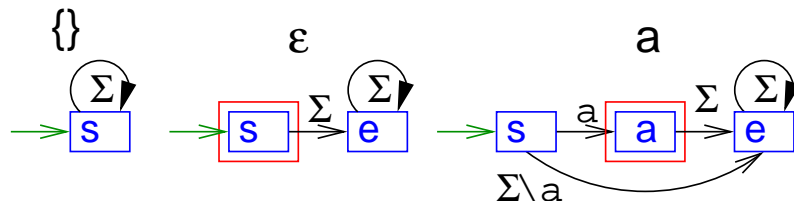
$P = s \xrightarrow{w} f$  mit  $w = a_1 \dots a_k$ .  $P$  ist mit  $w$  **beschriftet**

$q \xrightarrow{*} r$  es gibt Pfad von  $q$  nach  $r$ , d.h.,  $r$  ist von  $q$  aus **erreichbar**

Es gilt  $s \xrightarrow{*} s$  (Reflexivität)



### Endliche Automaten: Einfache Beispiele



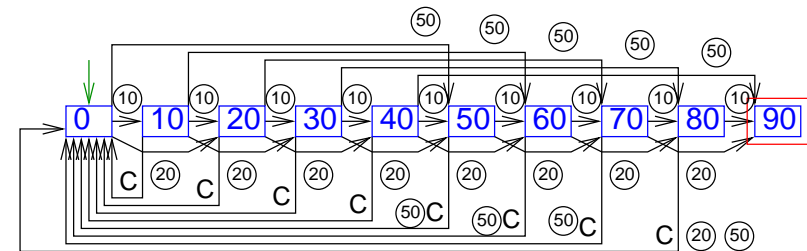
### Dualität: Grammatiken ↔ Maschinen

Grammatiken **erzeugen** Wörter.

Maschinen **akzeptieren/erkennen/parsen/verbrauchen** Wörter.



### Beispiel: Erfolgreiche Bezahlvorgänge





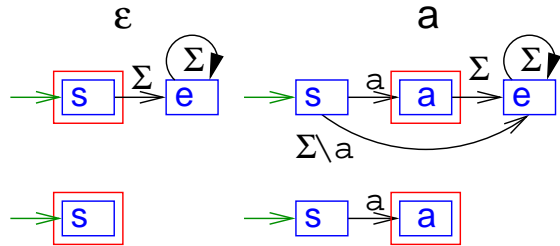
### Vervollständigung

Oft geben wir nicht alle Funktionswerte von  $\delta$  an.

Konvention: Es gibt immer einen Fehlerzustand  $e$  mit

$\delta(q, c) = e$  wenn kein anderer Wert angegeben.

$\delta(e, c) = e \forall c \in \Sigma$



### Satz:

### Durch DFAs erkennbare Sprachen sind vom Chomsky Typ3

Sei  $A = (Z, \Sigma, \delta, S, F)$  ein DFA.

Betrachte die Grammatik  $G = (Z, \Sigma, P, S)$  mit

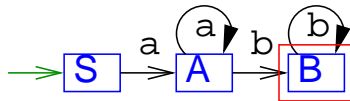
$$P = \{ Q \rightarrow aQ' : \delta(Q, a) = Q' \} \cup \\ \{ Q \rightarrow a : \delta(Q, a) = Q' \in F \} \cup \\ \{ S \rightarrow \varepsilon : S \in F \} .$$

z.Z.  $L(G) = L(A)$

(ggf.  $\varepsilon$  eliminieren...)



Beispiel:  $\{a^n b^m : n \geq 1, m \geq 1\}$



$G = (\{S, A, B\}, \{a, b\}, P, S)$

$q$	$c$	$\delta(q, c)$	$\in P$
$S$	$a$	$A$	$S \rightarrow aS$
$A$	$a$	$A$	$A \rightarrow aA$
$A$	$b$	$B$	$A \rightarrow bB, A \rightarrow b$
$B$	$b$	$B$	$B \rightarrow bB, B \rightarrow b$

Was ist mit  $\delta(S, b)$  ?



### Durch DFAs erkennbare Sprachen sind vom Chomsky Typ3

Sei  $A = (Z, \Sigma, \delta, S, F)$  ein DFA.

Betrachte die Grammatik  $G = (Z, \Sigma, P, S)$  mit

$$P = \{ Q \rightarrow aQ' : \delta(Q, a) = Q' \} \cup \\ \{ Q \rightarrow a : \delta(Q, a) = Q' \in F \} \cup \\ \{ S \rightarrow \varepsilon : S \in F \} .$$

z.Z.  $L(G) = L(A)$

Idee:  $\exists$  1-1 Beziehung zwischen

Ableitungen  $S \Rightarrow w_1 A_1 \Rightarrow w_1 w_2 A_2 \Rightarrow \dots \Rightarrow w_1 w_2 \dots w_{n-1} A_{n-1} \Rightarrow w$  und  
DFA Berechnungspfad  $S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \dots \xrightarrow{w_n} f \in F$ .



Beweis  $L(G) = L(A)$ :

**Fall**  $w = \varepsilon$ :  $\varepsilon \in L(G) \Leftrightarrow S \rightarrow \varepsilon \in P \Leftrightarrow S \in F \Leftrightarrow \varepsilon \in L(A)$

$A = (Z, \Sigma, \delta, S, F)$ ,  $G = (Z, \Sigma, P, S)$  mit  $P = \{Q \rightarrow aQ' : \delta(Q, a) = Q'\} \cup \{Q \rightarrow a : \delta(Q, a) = Q' \in F\} \cup \{S \rightarrow \varepsilon : S \in F\}$



**Beweis(skizze)**  $L(G) = L(A)$  **Fall**  $|w| = n, n > 0$ :

$w_1 \cdots w_n \in L(G)$

$\Leftrightarrow S \Rightarrow w_1 A_1 \Rightarrow w_1 w_2 A_2 \xrightarrow{*} w_1 \cdots w_{n-1} A_{n-1} \Rightarrow w_1 \cdots w_n$

$\Leftrightarrow \{S \rightarrow w_1 A_1, A_1 \rightarrow w_2 A_2, \dots, A_{n-2} \rightarrow w_{n-1} A_{n-1}, A_{n-1} \rightarrow w_n\} \subseteq P$

$\Leftrightarrow \delta(S, w_1) = A_1, \delta(A_1, w_2) = A_2, \dots, \delta(A_{n-1}, w_n) = A_n \in F$

$\Leftrightarrow \exists$  Berechnungspfad  $S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \cdots A_{n-1} \xrightarrow{w_n} A_n \in F$

$\Leftrightarrow w_1 \cdots w_n \in L(A)$

(In 2 Richtungen lesen. ' $\Leftrightarrow$ ' jeweils begründen) ■

$A = (Z, \Sigma, \delta, S, F)$ ,  $G = (Z, \Sigma, P, S)$  mit  $P = \{Q \rightarrow aQ' : \delta(Q, a) = Q'\} \cup \{Q \rightarrow a : \delta(Q, a) = Q' \in F\} \cup \{S \rightarrow \varepsilon : S \in F\}$



### 1.2.2 Nichtdeterministische (endliche) Automaten **NFA**

**Satz:** Chomsky Typ3 Sprachen sind durch endliche Automaten erkennbar.

(Umkehrung des Hauptsatzes von Abschnitt 1.2.1)

Problem: **Beweis umdrehbar ?**

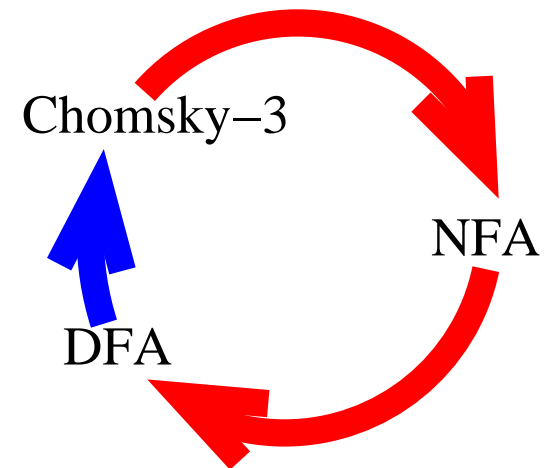
Nicht mit **DFA** !

z.B. ist aktuelle Variable

bei Chomsky-3 Ableitungen **nicht eindeutig** bestimmt.



### Der Plan



**Nichtdeterministische endliche Automaten NFA**

- mehrere erlaubte Übergänge für gegebenes Situation (Zustand, Eingabezeichen)

**Nichtdeterministischer endlicher Automat A**

- $Q$ , Zustandsmenge
- $\Sigma$ , Eingabealphabet
- $\delta: Q \times \Sigma \rightarrow 2^Q$ , Übergangsfunktion
- $s \in Q$ , Startzustand
- $F \subseteq Q$ , Endzustände

Zustandsübergang von  $q$  nach  $q'$  bei Eingabe von  $a: q' \in \delta(q, a)$   
i.allg. mehrere Möglichkeiten!

**Nichtdeterministischer endlicher Automat A**

Variante (äquivalent) —  $\delta$  als Relation.

- $Q$ , Zustandsmenge
- $\Sigma$ , Eingabealphabet
- $\delta \subseteq Q \times \Sigma \times Q$  Übergangsrelation
- $s \in Q$ , Startzustand
- $F \subseteq Q$ , Endzustände

A **kann** Zustandsübergang von  $q$  nach  $q'$  machen wenn  $a$  eingegeben wird und  $(q, a, q') \in \delta$ .

**Erweiterung von  $\delta$** 

**Zustandsmengen:**  $\bar{\delta}: 2^Q \times \Sigma \rightarrow 2^Q$

$$\bar{\delta}(M, a) := \bigcup_{p \in M} \delta(p, a)$$

**Zustandsmengen und Eingabeworte:**  $\hat{\delta}: 2^Q \times \Sigma^* \rightarrow 2^Q$

$$\hat{\delta}(M, \varepsilon) := M$$

$$\hat{\delta}(M, aw) := \hat{\delta}(\bar{\delta}(M, a), w)$$

$$L(A) := \{w \in \Sigma^* : \hat{\delta}(\{s\}, w) \cap F \neq \emptyset\}$$

**(Multi)Graphinterpretation für  $L(A)$** 

$$A = (Q, \Sigma, \delta, s, F)$$

$$G_A = (Q, E) \quad \text{Funktionsinterpretation von } \delta$$

mit  $e = (q, q') \in E$ , **Beschriftung**  $\ell(e) = a$  falls  $q' \in \delta(q, a)$

$$G_A = (Q, \delta) \quad \text{Relationeninterpretation von } \delta$$

schreibe Kanten  $e$  mit  $\ell(e) = a$  als  $(q, a, q')$ .

„Multi“=parallele Kanten erlaubt (unterschiedliche Beschriftungen)

$w \in L(A) \Leftrightarrow \exists$  Pfad  $P = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} f$  in  $A$  (in  $G(A)$ ):

$$f \in F \wedge w = a_1 a_2 \dots a_k$$

In Worten: Ein Pfad von  $s$  zu einem Endzustand ist mit  $w$  **beschriftet**.

**NFA  $\rightarrow$  DFA**

Gegeben: NFA  $A = (Q, \Sigma, \delta, s, F)$

**Satz: (Potenzmengenkonstruktion)**

[Rabin, Scott 1959, IBM J. of R&D]

DFA  $A' := (2^Q, \Sigma, \bar{\delta}, \{s\}, \{M \subseteq Q : M \cap F \neq \emptyset\})$  akzeptiert  $L(A)$ .

**Übung:** Entwerfen Sie einen Algorithmus, dessen Eingabe der NFA  $A$  und ein Wort  $w$  ist und der  $\hat{\delta}(\{s\}, w)$  in Zeit  $\mathcal{O}(|w| \cdot |\delta|)$  berechnet. Dabei ist  $|\delta|$  die Anzahl Einträge der Form  $p \in \delta(q, a)$ , die benötigt wird, um  $\delta$  zu definieren.



**Lemma:**  $\hat{\delta}(M, w) = \{q \in Q : \exists p \in M : p \xrightarrow{w} q\}$

**Beweis** durch Induktion über  $|w|$ :

$$\hat{\delta}(M, \varepsilon) = \{s\}$$

$$n \rightsquigarrow n + 1:$$

$$\hat{\delta}(M, aw) = \hat{\delta}(\bar{\delta}(M, a), w) \quad \text{(IV)}$$

$$= \{r \in Q : \exists q \in \bar{\delta}(M, a) : q \xrightarrow{w} r\} \quad \text{(Def. } \bar{\delta})$$

$$= \{r \in Q : \exists p \in M, q \in \delta(p, a) : q \xrightarrow{w} r\} \quad \text{(Graphint.)}$$

□

**Korollar:**  $L(A) = \{w \in \Sigma^* : \exists z \in F : s \xrightarrow{w} z\}$

**Potenzmengenkonstruktion**

$$A = (Q, \Sigma, \delta, s, F)$$

$$A' := (2^Q, \Sigma, \bar{\delta}, \{s\}, F'), \quad F' := \{M \subseteq Q : M \cap F \neq \emptyset\}$$

Behauptung:  $L(A') = L(A)$

Beweis:

$$L(A) = \{w \in \Sigma^* : \hat{\delta}(\{s\}, w) \cap F \neq \emptyset\} \quad \text{Def. } L(A)$$

$$= \{w \in \Sigma^* : \hat{\delta}(\{s\}, w) \in F'\} \quad \text{Def. } F'$$

$$= L(A') \quad \text{Def. } L(A')$$

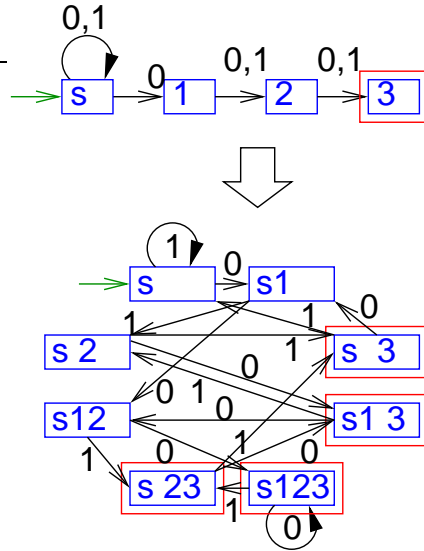
( $\hat{\delta}$  tanzt hier auf zwei Hochzeiten !)

□

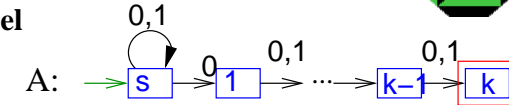


**Beispiel**

$q$	$\bar{\delta}(q, 0)$	$\bar{\delta}(q, 1)$
$s$	$s, 1$	$s$
$s, 1$	$s, 1, 2$	$s, 2$
$s, 2$	$s, 1, 3$	$s, 3$
$s, 3$	$s, 1$	$s$
$s, 1, 2$	$s, 1, 2, 3$	$s, 2, 3$
$s, 1, 3$	$s, 1, 2$	$s, 2$
$s, 2, 3$	$s, 1, 3$	$s, 3$
$s, 1, 2, 3$	$s, 1, 2, 3$	$s, 2, 3$



**Allgemeineres Beispiel**



**Satz:**  $\nexists$  DFA  $A' = (Q, \Sigma, \delta, s, F) : L(A') = L(A) \wedge |Q| < 2^k$

**Beweis:** Annahme:  $\exists A'$

$\rightarrow \exists x \neq y \in \{0, 1\}^k : \hat{\delta}(s, x) = \hat{\delta}(s, y)$  (Schubfachprinzip)

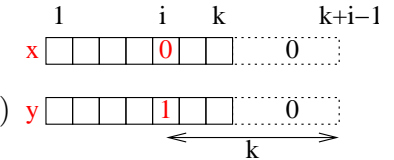
wähle  $i$  mit  $x[i] \neq y[i]$ ,

OBdA gilt  $x[i] = 0, y[i] = 1$

Also  $x0^{i-1} \in L(A)$

und  $y0^{i-1} \notin L(A)$ .

Aber,  $\hat{\delta}(s, x0^{i-1}) = \hat{\delta}(\hat{\delta}(s, x), 0^{i-1})$   
 $= \hat{\delta}(\hat{\delta}(s, y), 0^{i-1}) = \hat{\delta}(s, y0^{i-1})$ .



Also werden  $x0^{i-1}$  und  $y0^{i-1}$  entweder beide akzeptiert oder beide abgelehnt.

Widerspruch. □



**Implementierungshinweise**

Nur von  $\{s\}$  aus erreichbare Teilmengen erzeugen:

$Q' := \{\{s\}\}$

// states of  $A'$

Queue todo :=  $Q'$

**while**  $\exists M \in \text{todo}$  **do**

    todo := todo  $\setminus M$

**foreach**  $a \in \Sigma$  **do**

**if**  $M' = \bar{\delta}(M, a) \notin Q'$  **then**

            insert  $M'$  into  $Q'$

            insert  $M'$  into todo

Ofte  $|Q'| \ll 2^{|Q|}$



**Typ-3  $\rightarrow$  NFA**

Sei  $G = (V, \Sigma, P, S)$  eine Typ3 Grammatik.

Betrachte den NFA  $A = (V \cup \{f\}, \Sigma, \delta, S, \{f\} \cup \{S : S \rightarrow \epsilon \in P\})$

mit

$$\delta = \{(q, a, q') : q \rightarrow aq' \in P\} \cup \{(q, a, f) : q \rightarrow a \in P\}$$

(Relationennotation v.  $\delta$ ).

Es gibt eine 1-1 Beziehung zwischen Ableitungen der Form

$S \Rightarrow w_1 A_1 \Rightarrow w_1 w_2 A_2 \Rightarrow \dots \Rightarrow w_1 w_2 \dots w_{n-1} A_{n-1} \Rightarrow w$  in  $G$  und akzeptierenden Pfaden der Form

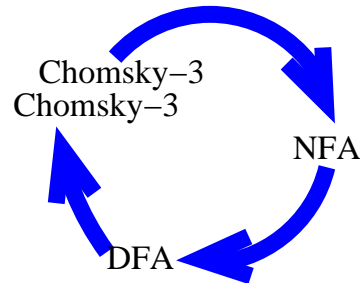
$S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \dots \xrightarrow{w_n} f$  durch  $A$ .

Also ist  $L(A) = L(G)$ . ■



### Mehrdeutigkeit bei Typ-3

**Satz:**  $\forall L \in \text{Typ-3} : \exists \text{Typ-3 Grammatik mit eindeutigen Ableitungen.}$



**Beweisskizze:** Sei  $A$  DFA mit  $L(A) = L$ .

Die zu  $A$  korrespondierende Typ-3 Grammatik hat eindeutige Ableitungen.



### 1.2.3 Reguläre Ausdrücke

Ein regulärer (?) Ausdruck **beschreibt** eine reguläre (?) Sprache.

Ausdruck	beschreibt	Bemerkung
$\emptyset$	$\emptyset$	
$\epsilon$	$\{\epsilon\}$	
$a$	$\{a\}$	$a \in \Sigma$
$\alpha \cup \beta$	$L(\alpha) \cup L(\beta)$	$\alpha$ beschreibt $L(\alpha)$ (Synonym: $\alpha \beta$ )
$\alpha \cdot \beta$	$L(\alpha) \cdot L(\beta)$	$\beta$ beschreibt $L(\beta)$
$(\alpha)$	$L(\alpha)$	
$\alpha^*$	$L(\alpha)^*$	
$\alpha^+$	$L(\alpha)^+$	

Nachlässigkeiten: '.' weglassen,  $L(\cdot)$  weglassen



### Syntax (reg. Ausdrücke) versus Semantik (reg. Sprachen)

Computerprogramme bearbeiten **syntaktische** Objekte.

### Programmverifikation

Wir beweisen auf **semantischer** Ebene, dass die Objekte korrekt verarbeitet werden.



### Beispiele

- vorletztes Zeichen 0:  $(0 \cup 1)^* 0 (0 \cup 1)$
- enthält 10:  $(0 \cup 1)^* 10 (0 \cup 1)^*$
- enthält nicht 10:  $0^* 1^*$
- enthält 101:  $(0 \cup 1)^* 101 (0 \cup 1)^*$
- enthält nicht 101:  $0^* 1^* \cup (0^* 1^* 100)^* 0^* 1^* 10 (\epsilon \cup 00^* 1^*)$
- ganze Zahl:  $(\epsilon \cup + \cup -) (1 \cup \dots \cup 9) (0 \cup \dots \cup 9)^* \cup 0$



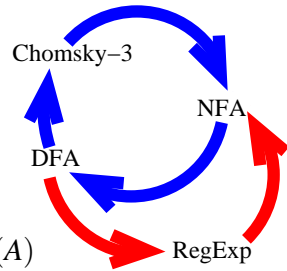
# Satz: Reguläre Ausdrücke $\Leftrightarrow$ Typ-3 Sprachen

## [Kleene]

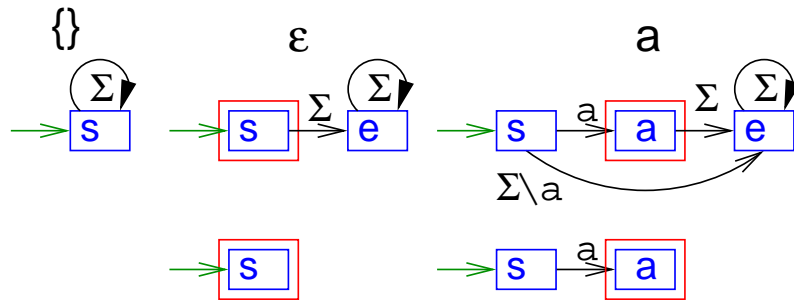
Der Plan

zu zeigen:

- regulärer Ausdruck  $\alpha$   
 $\rightsquigarrow$  NFA  $A$  mit  $L(\alpha) = L(A)$
- ← DFA  $A$   
 $\rightsquigarrow$  regulärer Ausdruck  $\alpha$  mit  $L(\alpha) = L(A)$



## Basisfälle



# Regulärer Ausdruck $\rightarrow$ NFA

Beweisidee:

baue Automaten für Teilausdrücke

stöpsle diese zu Automaten für komplexere Ausdrücke zusammen

Theoretikersprache:

**Strukturelle Induktion** über die Struktur eines regulären Ausdrucks.



## RegExp $\rightarrow$ NFA: $L_1 \cup L_2$

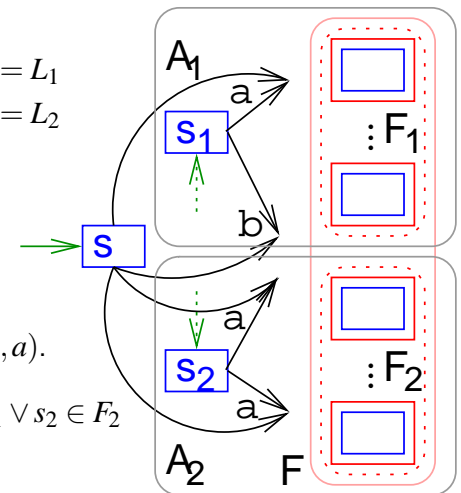
$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$   
 $A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$   
 sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (\{s\} \cup Q_1 \cup Q_2, \Sigma, \delta, s, F)$

$\delta$  verhält sich wie  $\delta_{1/2}$  für  $Q_{1/2}$

$\forall a \in \Sigma : \delta(s, a) := \delta(s_1, a) \cup \delta(s_2, a)$ .

$$F := \begin{cases} F_1 \cup F_2 \cup \{s\} & \text{falls } s_1 \in F_1 \vee s_2 \in F_2 \\ F_1 \cup F_2 & \text{sonst} \end{cases}$$





**Beweis von  $L_1 \cup L_2 \subseteq L(A)$**

Sei  $w \in L_1 = L(A_1)$  beliebig.

Fall  $w = \varepsilon$

$\rightarrow s_1 \in F_1 \rightarrow s \in F \rightarrow w \in L(A)$ .

Fall  $w = ax$ :

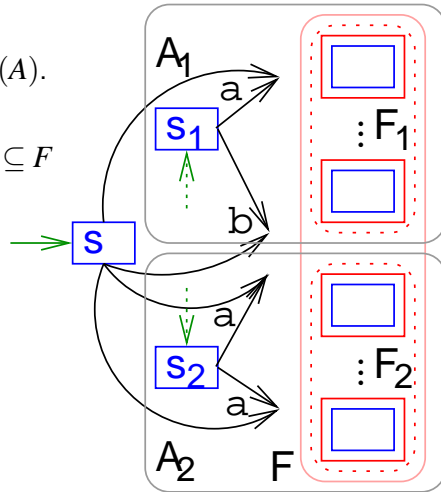
$\rightarrow \exists$  Pfad  $P_1 = s_1 \xrightarrow{a} q_1 \xrightarrow{x} f_1 \in F_1 \subseteq F$

$\rightarrow \exists$  Pfad  $P = s \xrightarrow{a} q_1 \xrightarrow{x} f_1 \in F$

$\rightarrow w \in L(A)$ .

$w \in L_2 = L(A_2)$

$\rightarrow \dots \rightarrow w \in L(A)$ .



**Beweis von  $L(A) \subseteq L_1 \cup L_2$**

Sei  $w \in L(A)$  beliebig.

Fall  $w = \varepsilon \rightarrow s \in F \rightarrow s_1 \in F_1 \vee s_2 \in F_2$

$\rightarrow \varepsilon \in L_1 \vee \varepsilon \in L_2 \rightarrow \varepsilon \in L_1 \cup L_2$

Fall  $w = ax$ :

$\rightarrow \exists$  Pfad  $P = s \xrightarrow{a} q \xrightarrow{x} f \in F$ .

Fall  $q = q_1 \in Q_1$ :

$\rightarrow \exists$  Pfad  $P_1 = s_1 \xrightarrow{a} q_1 \xrightarrow{x} f \in F_1$ .

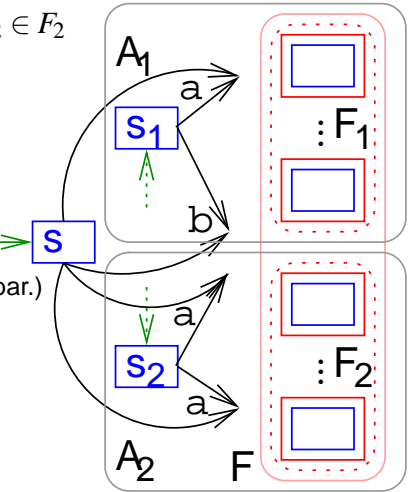
(von  $q_1$  sind nur Zustände in  $Q_1$  erreichbar.)

$\rightarrow ax = w \in L_1 \subseteq L_1 \cup L_2$

sonst:  $\rightarrow q = q_2 \in Q_2$

$\rightarrow \exists$  Pfad  $P_2 = s_2 \xrightarrow{a} q_2 \xrightarrow{x} f \in F_2$ .

$\rightarrow ax = w \in L_2 \subseteq L_1 \cup L_2$



$L_1 \cdot L_2$

$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$

$A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$

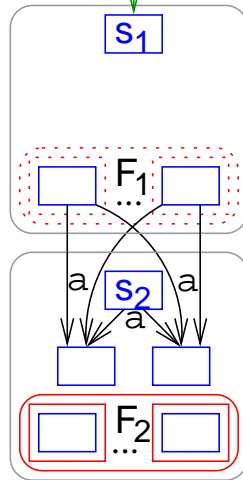
sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (Q_1 \cup Q_2, \Sigma, \delta, s_1, F)$ ,  $\forall a \in \Sigma$ :

$$\delta(q, a) := \begin{cases} \delta_1(q, a) & \text{falls } q \in Q_1 \setminus F_1 \\ \delta_1(q, a) \cup \delta_2(s_2, a) & \text{falls } q \in F_1 \\ \delta_2(q, a) & \text{sonst} \end{cases}$$

$$F := \begin{cases} F_1 \cup F_2 & \text{falls } s_2 \in F_2 \\ F_2 & \text{sonst} \end{cases}$$

Beweis: Übung

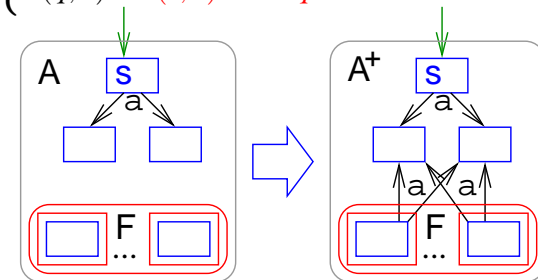


**Positive Hülle  $L^+ = \bigcup_{i \geq 1} L^i$**

$A = (Q, \Sigma, \delta, s, F)$  mit  $L(A) = L$

$A^+ := (Q, \Sigma, \delta^+, s, F)$ ,  $\forall a \in \Sigma$ :

$$\delta^+(q, a) := \begin{cases} \delta(q, a) & \text{falls } q \in Q \setminus F \\ \delta(q, a) \cup \delta(s, a) & \text{falls } q \in F \end{cases}$$





### Beweis von $L(A^+) \subseteq L^+$

Sei  $w \in L(A^+)$  beliebig (OBdA  $w \neq \varepsilon$ )

Sei  $P = s \xrightarrow{a_0} q_0 \xrightarrow{*} f$  ein akzeptierender Pfad für  $w$ .

Zerlege  $P$  an Übergängen der Form  $f_j \xrightarrow{a_j} q_j$

mit  $q_j \notin \delta(f_j, a_j)$ ,  $j \in 1..i$ ,  $i \geq 0$ .

$\rightarrow q_j \in F_j, q_j \in \delta(s, a_j)$ .

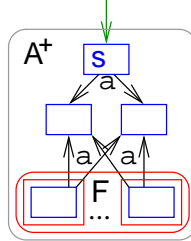
$$P = s \xrightarrow{a_0} q_0 \xrightarrow{x_0} f_1 \xrightarrow{a_1} q_1 \xrightarrow{x_1} f_2 \xrightarrow{*} f_i \xrightarrow{a_i} q_i \xrightarrow{x_i} f$$

Definiere  $P_j := s \xrightarrow{a_j} q_j \xrightarrow{x_j} f_{j+1}$  (mit  $f_{i+1} := f$ ).

$\rightarrow \forall j \in 0..i : P_j$  ist akzeptierender Pfad in  $A$ .

$\rightarrow w \in L^+$

□



### Beweis von $L^i \subseteq L(A^+)$ für $i \geq 1$

Sei  $w = w_1 \cdots w_i \in L^i$  beliebig (OBdA  $\varepsilon \neq w_i \in L$ ).

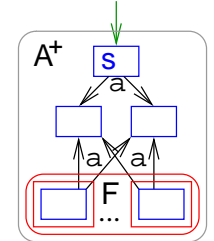
Betrachte Pfade  $P_j = s \xrightarrow{a_j} q_j \xrightarrow{x_j} f_j$ ,  $j \in 1..i$ ,  $f_j \in F$ , die  $w_1 \in L, \dots, w_i \in L$  bezeugen.

$$\xrightarrow{P} s \xrightarrow{a_1} q_1 \xrightarrow{x_1} f_1 \xrightarrow{a_2} q_2 \xrightarrow{x_2} f_2 \xrightarrow{*} f_{i-1} \xrightarrow{a_i} q_i \xrightarrow{x_i} f_i$$

ist Pfad in  $A^+$ , der  $w \in L(A^+)$  bezeugt.

$\rightarrow w \in L(A^+)$

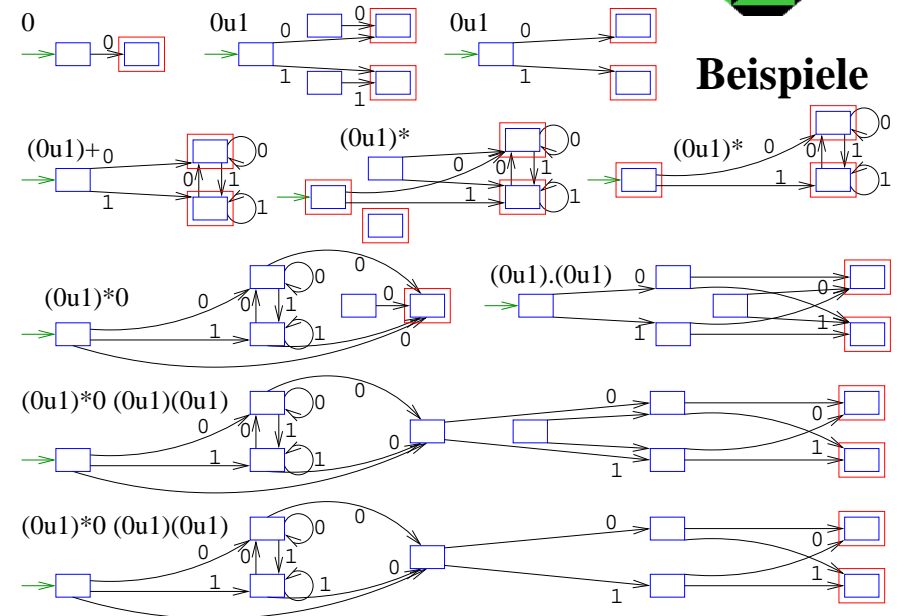
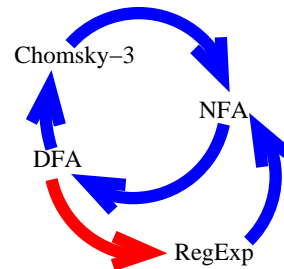
□



### Kleensche Hülle $L^*$

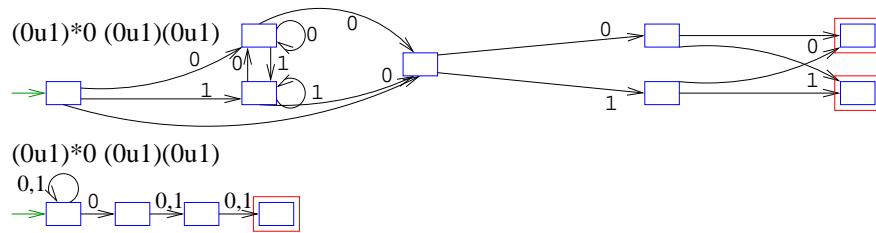
Baue Automaten für  $\varepsilon \cup L^+ = L^*$ .

■





## Beispiele



## Exkurs: Anwendung Texte durchsuchen

Unix-Tool `grep:grep REGULAR-EXPRESSION FILE`

- Suche all Teilstrings in FILE, die in  $L(\text{REGULAR-EXPRESSION})$  sind
- Viel syntaktischer Zuckerguß: Bereiche a-g, vordefinierte Ausdrücke z.B. `:alnum:`, vorgegebene Anzahl Wiederholungen,...
- Aber alles ist leicht in einfache reg. Ausdrücke übersetzbar.
- Schnelle Ausführung durch Übersetzung in deterministische endliche Automaten.



## Anwendung Scanner-(Generatoren), `lex`, `flex`

**Eingabe:** Ein System von regulären Ausdrücken

**Ausgabe:** Ein endlicher Automat (C code),

**Laufzeit-Eingabe:** Das Programm als **Zeichenkette**

**Laufzeit-Ausgabe:** Das Programm als Folge von **Token** (Pakete) wie Zahl, Bezeichner, Schlüsselwort.

- zeitkritisch**, weil nur hier jedes einzelne Zeichen angefasst wird, Kommentare weggeworfen werden, Dezimalzahlen binär gewandelt werden,...
- normiert** die Darstellung, durch Entfernen von Leerzeichen Varianten von Zahlrepräsentationen,...
- vereinfacht die nachfolgende Syntaxanalyse



## Ähnliche Funktionalität

- Editoren**, z.B. emacs
- Skript-Sprachen** wie Perl (berüchtigt?)
- `java.util.regex` Bibliothek
- C++** Boost.Regex Bibliothek
- .net framework
- Vorverarbeitung beim Parsing von **xml** Dokumenten

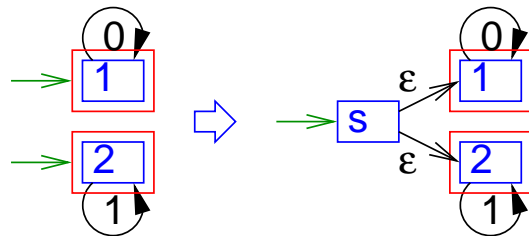


## Exkurs: $\epsilon$ -Übergänge

wir erlauben **spontane** Übergänge  
ohne Verbrauch eines Eingabezeichens.



## Beispiel: $0^* \cup 1^*$



## $\epsilon$ NFA

- $Q$ , Zustandsmenge
- $\Sigma$ , Eingabealphabet
- $\delta: Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$ , Übergangsfunktion
- $s \in Q$ , Startzustand
- $F \subseteq Q$ , Endzustände



## RegExp $\rightarrow$ $\epsilon$ NEA: $L_1 \cup L_2$

$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$

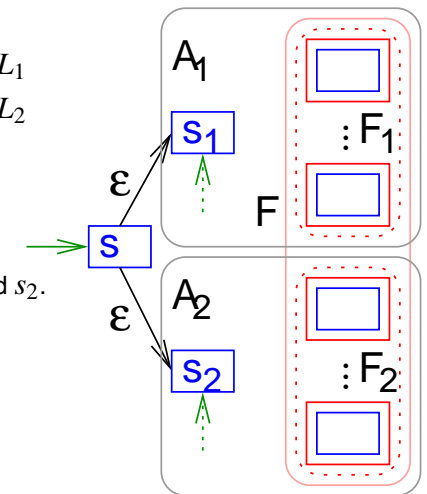
$A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$

sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (\{s\} \cup Q_1 \cup Q_2, \Sigma, \delta, s, F_1 \cup F_2)$

$\delta$  verhält sich wie  $\delta_{1/2}$  für  $Q_{1/2}$

spontane Übergänge von  $s$  nach  $s_1$  und  $s_2$ .





$$L_1 \cdot L_2$$

$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$

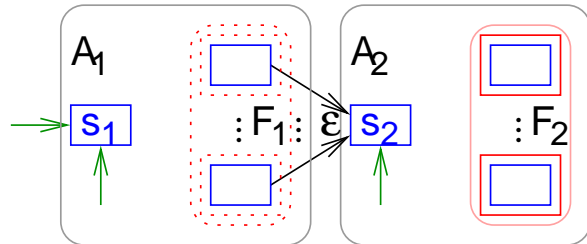
$A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$

sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (Q_1 \cup Q_2, \Sigma, \delta, s_1, F_2)$

$\delta$  verhält sich wie  $\delta_{1/2}$  für  $Q_{1/2}$

spontane Übergänge von  $F_1$  nach  $s_2$ .



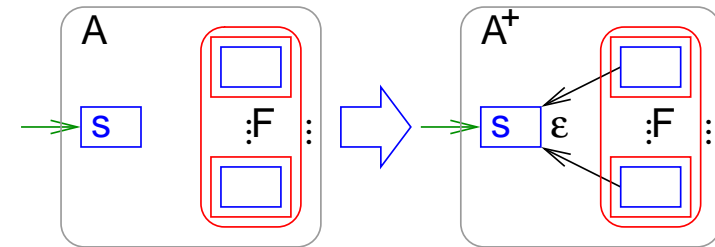
**Positive Hülle**  $L^+ = \bigcup_{i \geq 1} L^i$

$A = (Q, \Sigma, \delta, s, F)$  mit  $L(A) = L$

$A^+ := (Q, \Sigma, \delta^+, s, F)$

$\delta^+$  verhält sich wie  $\delta$

spontane Übergänge  $f \rightarrow s \forall f \in F$ .



$$\epsilon\text{NFA } A \rightsquigarrow \text{NFA } \bar{A}$$

**Beweisidee:**

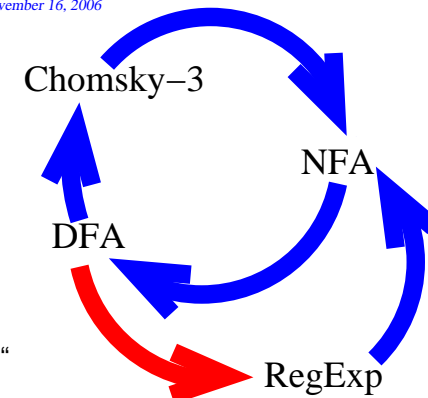
ersetze mit  $\epsilon$  und einem Buchstaben beschriftete Pfade in  $A$  durch explizite Übergänge in  $\bar{A}$ .

Vorsicht mit Endzuständen.



**DFA  $\rightarrow$  RegExp**

Ziel:



- Kein „Sprachenzoo“
- reguläre Ausdrücke sind universelles Interface
- reverse engineering (vielleicht)



**Beweis:** geg: DFA  $A = (\{1, \dots, n\}, \Sigma, \delta, s, F)$

ges: regulärer Ausdruck  $\alpha$  mit  $L(A) = L(\alpha)$ .

Für  $f \in F$  sei  $L_f = \{w \in \Sigma^* : \hat{\delta}(s, w) = f\}$ .

Da  $L(A) = \bigcup_{f \in F} L_f$ , genügt es, einen RegExp für  $L_f$  zu finden.



geg: DFA  $A_f = (\{1, \dots, n\}, \Sigma, \delta, s, \{f\})$

ges: regulärer Ausdruck  $\alpha$  mit  $L_f = L(A_f) = L(\alpha)$ .

Sei  $L_{ij} := L(\{1, \dots, n\}, \Sigma, \delta, i, \{j\})$

Also insbesondere  $L_{sf} = L_f$ .

$L_{ij}^m := \{w \in \Sigma^* : \exists \text{Abarbeitungspfad } i \xrightarrow{w} j = iPj \text{ mit } P \in \{1, \dots, m\}^*\}$

Beachte, dass  $L_{ij} = L_{ij}^n$ .

Wir konstruieren  $L_{ij}^m$  induktiv

aus den regulären Ausdrücken für kleinere  $m$ .

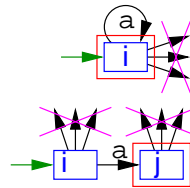


$L_{ij}^m := \{w \in \Sigma^* : \exists \text{Abarbeitungspfad } i \xrightarrow{w} j = iPj \text{ mit } P \in \{1, \dots, m\}^*\}$

geg: reguläre Ausdrücke  $\alpha_{ij}^k, k < m$  mit  $L(\alpha_{ij}^k) = L_{ij}^k$

ges:  $\alpha_{ij}^m$  mit  $L(\alpha_{ij}^m) = L_{ij}^m$

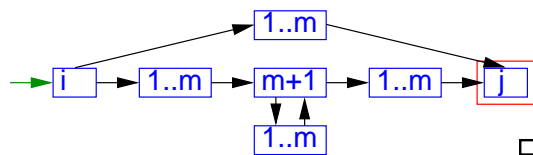
**Fall  $m = 0, i = j$ :**  $\alpha_{ii}^0 = \bigcup_{a \in \Sigma: \delta(i,a)=i} a \cup \epsilon$



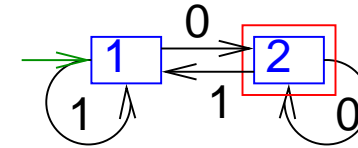
**Fall  $m = 0, i \neq j$ :**  $\alpha_{ij}^0 = \bigcup_{a \in \Sigma: \delta(i,a)=j} a$

**Fall  $m \rightsquigarrow m + 1$ :**

$\alpha_{ij}^{m+1} = \alpha_{ij}^m \cup \alpha_{i,m+1}^m \cdot (\alpha_{m+1,m+1}^m)^* \cdot \alpha_{m+1,j}^m$



**Beispiel**



$$\begin{aligned} \alpha_{11}^0 &= 1 \cup \epsilon & \alpha_{22}^0 &= 0 \cup \epsilon & \alpha_{12}^0 &= 0 & \alpha_{21}^0 &= 1 \\ \alpha_{12}^1 &= \alpha_{12}^0 \cup \alpha_{11}^0 \cdot (\alpha_{11}^0)^* \cdot \alpha_{12}^0 & \alpha_{22}^1 &= \alpha_{22}^0 \cup \alpha_{21}^0 \cdot (\alpha_{11}^0)^* \cdot \alpha_{12}^0 \\ &= 0 \cup (1 \cup \epsilon) \cdot (1 \cup \epsilon)^* \cdot 0 & &= 0 \cup \epsilon \cup 1 \cdot (1 \cup \epsilon)^* \cdot 0 \\ &= 1^* 0 & &= 1^* 0 \cup \epsilon \end{aligned}$$

$$\begin{aligned} \alpha_{12}^2 &= \alpha_{12}^1 \cup \alpha_{12}^1 \cdot (\alpha_{22}^1)^* \cdot \alpha_{12}^1 \\ &= 1^* 0 \cup 1^* 0 \cdot (1^* 0 \cup \epsilon)^* \cdot (1^* 0 \cup \epsilon) \\ &= 1^* 0 (1^* 0)^* \end{aligned}$$

$L(\alpha_{12}^2) = L_{12}^2 = L_{12} = L_2$ , wobei  $F = \{2\}$ .



### 1.2.4 Das Pumping Lemma (für reguläre Sprachen)

$L$  regulär

$$\longrightarrow \exists n \in \mathbb{N} : \forall w \in L : |w| > n$$

$$\longrightarrow \exists u, v, x : w = uvx \wedge$$

1.  $|v| \geq 1 \wedge$
2.  $|uv| \leq n \wedge$
3.  $\forall i \in \mathbb{N}_0 : uv^i x \in L$

In Worten:

Hinreichend lange Worte einer regulären Sprache lassen sich durch

**Wiederholung** eines nichttrivialen **Mittelteiles** „aufpumpen“.



### Beweis Pumping Lemma

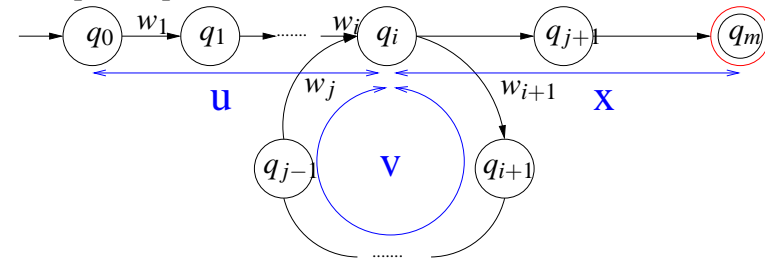
$L$  regulär  $\longrightarrow \exists n \in \mathbb{N} : \forall w \in \Sigma^* : |w| > n \longrightarrow \exists u, v, x :$

$$w = uvx \wedge |v| \geq 1 \wedge |uv| \leq n \wedge \forall i \in \mathbb{N}_0 : uv^i x \in L$$

**Beweis:** Sei  $A = (Q, \Sigma, \delta, q_0, F)$  DFA mit  $L(A) = L$ .

Sei  $n = |Q|$  und  $w \in L$  mit  $|w| = m > n$  beliebig.

Seien  $q_0, \dots, q_m$  die durchlaufenen Zustände.



$(\exists i < j \leq n : q_i = q_j) \longrightarrow |v| \geq 1, |uv| \leq n, uv^i x$  wird ebenfalls akzeptiert



**Beispiel:**  $L = \{a^k b^k : k \in \mathbb{N}\}$

Annahme  $L$  regulär.

Sei  $n$  der Wert aus dem Pumping Lemma und betrachte

$w = a^n b^n = uvx$  so dass laut Pumping Lemma  $ux \in L$ .

$|uv| \leq n, |v| \geq 1 \longrightarrow v = a^\ell$  für ein  $\ell \geq 1$ .

$ux = a^{n-\ell} b^n \in L$ .

Widerspruch. ■



**Beispiel: Wohlgeformte Klammerausdrücke**  $L_{()}()$

Annahme  $L$  regulär.

Sei  $n$  der Wert aus dem Pumping Lemma und betrachte

$w = ({}^n) = uvx$  so dass laut Pumping Lemma

$ux \in L_{()}()$  sowie  $|v| > 1$  und  $|uv| \leq n$ .

Dann ist  $v = ({}^i)$

und  $ux = ({}^{n-i}) \notin L_{()}()$  Widerspruch.



$$L = \{0^p : p \text{ ist Primzahl}\}$$

Annahme  $L$  regulär.

Sei  $n$  der Wert aus dem Pumping Lemma.

Sei  $p \geq n + 2$  Primzahl. ( $\exists$  unendlich viele Primzahlen)

$$\rightarrow 0^p \in L = uvw, |v| \geq 1, |uw| \geq 2.$$

Pumping-Lemma:  $uv^{|uw|}w \in L$ .

$$\rightarrow |uw| + |uw| \cdot |v| = |uw|(1 + |v|) \text{ ist Primzahl.}$$

Zwei nichttriviale Faktoren  $|uw| \geq 2$  und  $(1 + |v|) \geq 2$ .

Widerspruch. ■



## Das Pumping-Lemma ist keine hinreichende Bedingung für Regulärität

**Beispiel:**  $L = \{c^m a^\ell b^\ell : m, \ell \geq 0\} \cup \{a, b\}^*$  ist nicht regulär, aber

Sei  $n \geq 1$ ,  $x \in L$  beliebig mit  $|x| \geq n$ .

**Fall**  $x \in a^* b^*$ :

$$\text{Schreibe } x = \underbrace{\varepsilon}_u \underbrace{a}_v \underbrace{a^m b^{n-m-1}}_w$$

1.  $|v| = 1 \geq 1$
2.  $|uv| = 1 \leq n$
3.  $uv^i w = a^i a^m b^{n-m-1} \in a^* b^* \subseteq L$



## Das Pumping-Lemma ist keine hinreichende Bedingung für Regulärität

**Beispiel:**  $L = \{c^m a^\ell b^\ell : m, \ell \geq 0\} \cup \{a, b\}^*$  ist nicht regulär, aber

Sei  $n$  (sogar) beliebig,  $w \in L$  beliebig mit  $|w| \geq n$ .

**Fall**  $w \in a^* b^*$ : Kein Problem.

**Fall**  $w = a^m a^\ell b^\ell$ ,  $m \geq 1$ :

$$\text{Schreibe } w = \underbrace{\varepsilon}_u \underbrace{c}_v \underbrace{c^{m-1} a^\ell b^\ell}_x$$

1.  $|v| = 1 \geq 1$
2.  $|uv| = 1 \leq n$
3.  $uv^i x = c^{m-1+i} a^\ell b^\ell \in L$



### 1.2.5 Äquivalenzrelationen und Minimalautomaten

Idee: arbeite direkt mit  $L$  ohne Bezug zu einem konkreten Automaten.

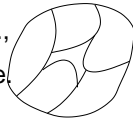


## Zur Erinnerung: Äquivalenzrelationen

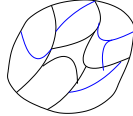
Eine Relation  $R \subseteq Y \times Y$  heisst Äquivalenzrelationen gdw. sie:

- reflexiv  $\forall x : xRx$
- transitiv  $\forall xyz : xRy \wedge yRz \longrightarrow xRz$
- symmetrisch ist.  $\forall xy : xRy \longrightarrow yRx$

**Äquivalenzklasse:**  $[x] = \{y : xRy\}$ . Äq.-Klassen sind gleich oder disjunkt und induzieren eine Partitionierung von  $Y$ , d.h., jedes Element von  $Y$  gehört zu genau einer Äq.-Klasse.



**Index:**  $\text{index}(R) := |\text{Äquivalenzklassen}| = |\{[x] : x \in Y\}|$



## Verfeinerung: $R$ verfeinert $R'$ gdw $R \subseteq R'$

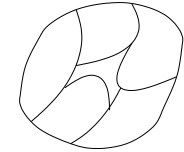
**Lemma:**  $R$  verfeinert  $R' \longrightarrow \forall$  Äquivalenzklassen  $[x]_R : [x]_R \subseteq [x]_{R'}$

**Beweis:**

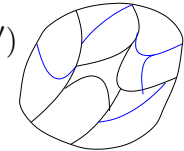
$$y \in [x]_R \Leftrightarrow (y, x) \in R$$

$$\xrightarrow{R \subseteq R'} (y, x) \in R'$$

$$\Leftrightarrow y \in [x]_{R'}$$



**Korollar:**  $R$  verfeinert  $R' \longrightarrow \text{index}(R) \geq \text{index}(R')$



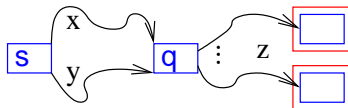
## Nerode Relation

Zur Sprache  $L$  ist die **Nerode Relation**

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

Idee: Äquivalenzklassen entsprechen Zuständen.

Wieso? Genauer!



## DFAs induzieren Äquivalenzrelationen

Sei  $M = (Q, \Sigma, \delta, s, F)$  DFA mit  $L(M) = L$ .

$$R_M := \{(x, y) \in \Sigma^* \times \Sigma^* : \hat{\delta}(s, x) = \hat{\delta}(s, y)\}$$

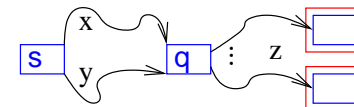
Äquivalenzrelation! Eine Äquivalenzklasse pro (von  $s$  erreichbarem) Zustand.

**Lemma 1:**  $R_M$  verfeinert die Neroderelation  $R_L =$

$$\{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

**Beweisskizze:** z.Z.

$$\forall (x, y) \in \Sigma^* \times \Sigma^* : \hat{\delta}(s, x) = \hat{\delta}(s, y) \longrightarrow \forall z : xz \in L \Leftrightarrow yz \in L$$





## Unendlicher Index der Neroderelation

**Beobachtung:**  $\text{index}(R_L) = \infty \longrightarrow L$  ist nicht regulär.

**Beweis:** Annahme  $L$  ist regulär.

$\longrightarrow \exists$  DFA  $M = (Q, \Sigma, \delta, s, F) : L(M) = L$ .

$\longrightarrow R_M$  verfeinert  $R_L$ .

$\longrightarrow |Q| \geq \text{index}(R_M) \geq \text{index}(R_L) = \infty$ .

Widerspruch

**Ab jetzt:**  $\text{index}(R_L) < \infty$ .



## Äquivalenzklassenautomat

$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$

$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv})$  mit

$F_{\equiv} := \{[w] : w \in L\}$  und

$\delta_{\equiv}([w], a) := [wa]$ .

**Lemma:**  $\delta_{\equiv}$  ist wohldefiniert

Also  $xR_L y \longrightarrow \forall a \in \Sigma : xaR_L ya$

$xR_L y \longrightarrow \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L$

$\longrightarrow \forall az \in \Sigma^* : x(az) \in L \Leftrightarrow y(az) \in L$

$\Leftrightarrow \forall a \in \Sigma : \forall z \in \Sigma^* : (xa)z \in L \Leftrightarrow (ya)z \in L$

$\longrightarrow \forall a \in \Sigma : xaR_L ya$

*Rechtsinvarianz*

insbesondere

□



## Äquivalenzklassenautomat

$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$

Idee: wenn die Äquivalenzklassen  $[w_1], \dots, [w_k]$  von  $R_L$  zu den

Zuständen eines DFA  $M_{\equiv}$  korrespondieren, so ist dieser wegen

Lemma 1 DER **zustandsminimale** Automat für  $L$ .

$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv})$  mit

$F_{\equiv} := \{[w] : w \in L\}$  und

$\delta_{\equiv}([w], a) := [wa]$ .

**Lemma:**  $\delta_{\equiv}$  ist wohldefiniert

**Lemma:**  $\hat{\delta}_{\equiv}([\varepsilon], w) = [w]$

**Lemma:**  $L(M_{\equiv}) = L$



## Äquivalenzklassenautomat

$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$

$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv})$  mit

$F_{\equiv} := \{[w] : w \in L\}$  und

$\delta_{\equiv}([w], a) := [wa]$ .

**Lemma:**  $\hat{\delta}_{\equiv}([x], y) = [xy]$

Induktion über  $|y|$ :

$\hat{\delta}_{\equiv}([x], \varepsilon) = [x]$ .

$\hat{\delta}_{\equiv}([x], aw) \stackrel{\text{Def. } \hat{\delta}_{\equiv}}{=} \hat{\delta}_{\equiv}(\delta_{\equiv}([x], a), w) \stackrel{\text{Def. } \hat{\delta}_{\equiv}}{=} \delta_{\equiv}([xa], w) \stackrel{IV}{=} [xaw]$ . □



### Äquivalenzklassenautomat: akzeptiert $L$

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}) \text{ mit}$$

$$F_{\equiv} := \{[w] : w \in L\} \text{ und}$$

$$\delta_{\equiv}([w], a) := [wa].$$

**Lemma:**  $L(M_{\equiv}) = L$ .

$$w \in L(M_{\equiv})$$

$$\Leftrightarrow \hat{\delta}_{\equiv}([\varepsilon], w) \in \{[w] : w \in L\}$$

Def.  $M_{\equiv}$

$$\Leftrightarrow [w] \in \{[w] : w \in L\}$$

voriges Lemma

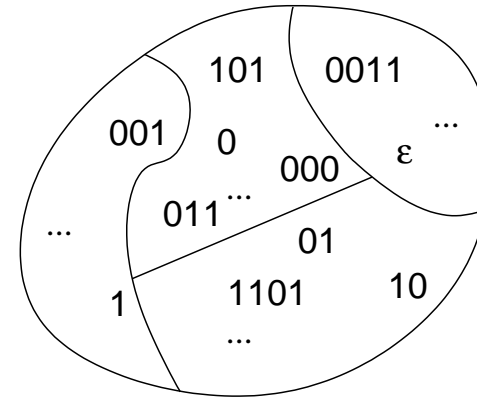
$$\Leftrightarrow w \in L.$$

Äq.Klassen sind ganz oder garnicht in  $L$



### Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen

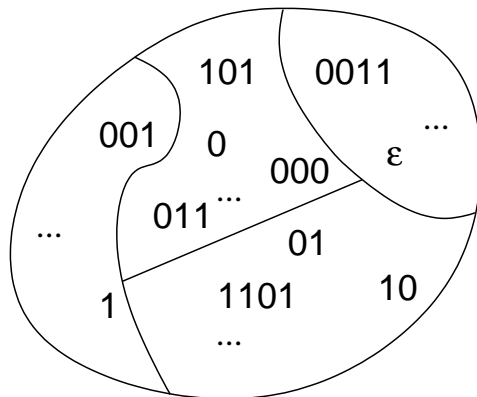


Äquivalenzklassen?



### Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen



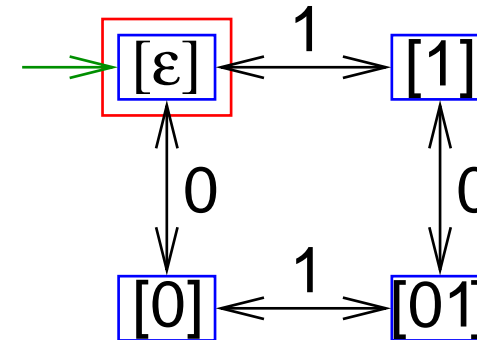
Äquivalenzklassen:

$$[\varepsilon], [0], [1], [01]$$



### Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen





## Satz von Nerode

Sei

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}.$$

$L$  nicht regulär  $\rightarrow \text{index}(R_L) = \infty$

$L$  regulär  $\rightarrow$  Sei

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}), \text{ und}$$

$$M = (Q, \Sigma, \delta, s, F) \text{ beliebiger DFA mit } L(M) = L.$$

Es gilt  $L(M_{\equiv}) = L$  und  $R_M$  ist Verfeinerung von  $R_L$ .

### Korollar:

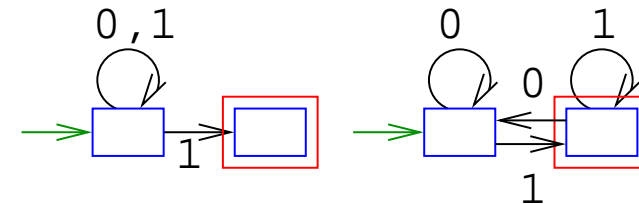
Alle zustandsminimalen Automaten für  $L$  sind isomorph zu  $M_{\equiv}$ .

(gleich bis auf Zustandsumbenennung)



## Gegenbeispiel NFA

Es gibt **strukturell unterschiedliche zustandsminimale** NFAs für  $(0 \cup 1)^*1$ .



Übung: Übergangsfunktionen hinschreiben



## Konstruktion eines zustandsminimalen Automaten

### Vorverarbeitung

Entferne Zustände, die von  $s$  aus **nicht erreichbar** sind.

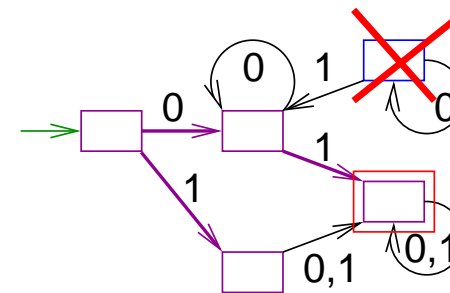
Algorithmus: **Tiefensuche** im Graph  $G_A$  von  $s$  aus.

Markiere alle erreichten Zustände.

Entferne nicht erreichte Zustände.



## Vorverarbeitung — Beispiel



Kante im  
Tiefensuchbaum

erreichter Zustand



## Äquivalente Zustände

Idee: betrachte DFA  $M = (Q, \Sigma, \delta, s, F)$  (ohne nichterr. Zustände)

$M$  nicht zustandsminimal  $\rightarrow$

$R_M$  verfeinert  $R_L \rightarrow$

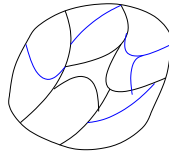
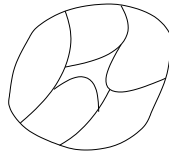
$\exists q \neq r \in Q : \text{Äq-Klasse}(q) \cup \text{Äq-Klasse}(r) \subseteq K$

für eine Äq-Klasse  $K$  von  $R_L$

solche  $q, r$  nennen wir **äquivalent**,  $q \equiv r$ .

Insbesondere:

$\forall w \in \Sigma^* : \hat{\delta}(q, w) \in F \Leftrightarrow \hat{\delta}(r, w) \in F$



## Entfernen äquivalenter Zustände

Betrachte  $q \neq r \in Q : q \equiv r$  mit  $r \neq s$

Entferne  $r$ :

$M' := (Q \setminus \{r\}, \Sigma, \delta', s, F \setminus \{r\})$  mit

$$\delta'(t, a) := \begin{cases} q & \text{falls } \delta(t, a) = r \\ \delta(t, a) & \text{sonst} \end{cases}$$

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

Beweis: Übung



## Zustandsminimierung

Erster Versuch:

**Function** minDFA( $M$ )

remove states not reachable from  $s$

**while**  $\exists q, r \in Q : q \equiv r \wedge q \neq r \wedge q \neq s$  **do**

remove  $q$  from  $M$

**return**  $M$

**Problem:** wie findet man äquivalente Zustände?

$q \equiv r$  gdw.  $\forall z \in \Sigma^* : \hat{\delta}(q, z) \in F \Leftrightarrow \hat{\delta}(r, z) \in F$

Allquantifizierung über **unendliche** Menge!

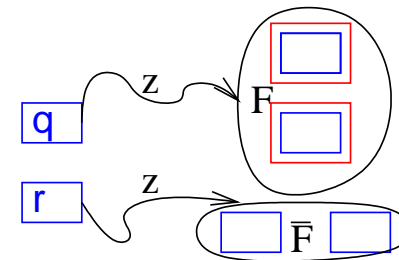


## Nichtäquivalenz von Zuständen

$q \equiv r$  gdw.  $\forall z \in \Sigma^* : \hat{\delta}(q, z) \in F \Leftrightarrow \hat{\delta}(r, z) \in F$

$q \not\equiv r$  gdw.  $\exists z \in \Sigma^* : \hat{\delta}(q, z) \in F \not\Leftrightarrow \hat{\delta}(r, z) \in F$

$z$  **bezeugt** Nichtäquivalenz.



Problem: finde Zeugen für Nichtäquivalenz



### Kürzeste Zeugen für Nichtäquivalenz

$\forall q \in F, r \notin F : \varepsilon$  bezeugt  $q \not\equiv r$ .

Sei  $w = aw'$  kürzester Zeuge für  $q \not\equiv r$ .

**Beobachtung:**  $w'$  ist Zeuge für  $q' := \delta(q, a) \not\equiv \delta(r, a) =: r'$

**Lemma:**  $w'$  ist kürzester Zeuge für  $q' \not\equiv r'$

Widerspruchsbeweis: Annahme:

$w''$  ist kürzerer Zeuge für  $q' \not\equiv r'$

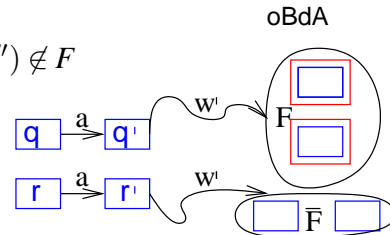
$\rightarrow \hat{\delta}(q', w'') \in F \wedge \hat{\delta}(r', w'') \notin F$

$\rightarrow \hat{\delta}(\delta(q, a), w'') \in F \wedge \hat{\delta}(\delta(r, a), w'') \notin F$

$\rightarrow \hat{\delta}(q, aw'') \in F \wedge \hat{\delta}(r, aw'') \notin F$

$\rightarrow aw''$  ist kürzerer Zeuge für  $q \not\equiv r$

Widerspruch



### Kürzeste Zeugen für Nichtäquivalenz

$\varepsilon$  bezeugt  $q \not\equiv r$  falls  $q \in F, r \notin F$ .

Sei  $w = aw'$  kürzester Zeuge für  $q \not\equiv r$ .

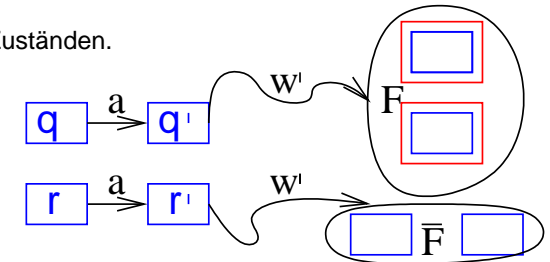
**Lemma:**  $w'$  ist kürzester Zeuge für  $q' := \delta(q, a) \not\equiv \delta(r, a) =: r'$

**Folgerung:**

kürzeste Zeugen lassen sich systematisch und **effizient** erzeugen.

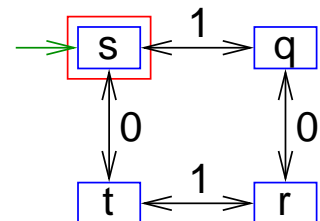
$\rightsquigarrow$  alle Nichtäquivalenzen

$\rightsquigarrow$  Äquivalenzklassen von Zuständen.



### Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen



$0 = 0\varepsilon$  ist kürzester Zeuge für  $t \not\equiv r$ .

$\rightsquigarrow \varepsilon$  ist kürzester Zeuge

für  $s = \delta(t, 0) \not\equiv \delta(r, 0) = q$ .



Sei  $w = aw'$  kürzester Zeuge für  $q \not\equiv r$ .

**Lemma:**  $w'$  ist kürzester Zeuge für  $q' := \delta(q, a) \not\equiv \delta(r, a) =: r'$

Sei  $N$  die Menge aller nichtäquivalenten Zustandspaare für die es Zeugen der Länge  $\leq k$  gibt.

Für welche Zustandspaare  $\{q, r\}$  gibt es Zeugen der Länge  $k + 1$ ?

**Lemma:**  $\{\{q, r\} \subseteq Q : \exists a \in \Sigma : \{\delta(q, a), \delta(r, a)\} \in N\}$



## Ein Einfacher Algorithmus

$N := \emptyset$  // marked pairs  
 $N' := \{\{q, r\} \subseteq Q : q \in F \not\leftrightarrow r \in F\}$  // pairs to be marked next  
**while**  $N' \neq \emptyset$  **do**  
      $N := N \cup N'$   
      $N' := \{\{q, r\} \subseteq Q : \exists a \in \Sigma : \{\delta(q, a), \delta(r, a)\} \in N\} \setminus N$

**Gesamtzeit:**  $\mathcal{O}(|\Sigma| \cdot |Q|^3)$

Initialisierung:  $\mathcal{O}(|Q|^2)$

Ein Schleifendurchlauf:  $\mathcal{O}(|\Sigma| \cdot |Q|^2)$

Wieviel **Schleifendurchläufe**? Sicherlich  $\leq |Q|^2$ .

Genauere Betrachtung:  $\leq |Q|$  **Schleifendurchläufe**



## Wieso Zustandsminimierung

- Minimiert Platz für **Zustandsübergangstabelle**
- Minimalautomat **eindeutig**  $\rightsquigarrow$  wir lernen etwas über die akzeptierte Sprache.

**Aber**, wenn  $\delta$  als **Schaltkreis** oder **Programm** repräsentiert ist, wollen wir deren Größe und Zeitaufwand optimieren.

Im allgemeinen schwierig  $\rightsquigarrow$  aktives Forschungsgebiet



## Zustandsminimierung in Zeit $\mathcal{O}(|\Sigma| \cdot |Q| \log |Q|)$

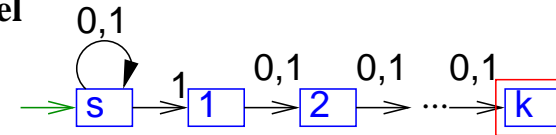
[Hopcroft 1971]. Geschickte Datenstrukturen.

Leichte Vereinfachung:

[Blum, Minimization of finite automata in  $\mathcal{O}(n \log n)$  time, Inf. Proc. Letters, 1996.]



## Beispiel

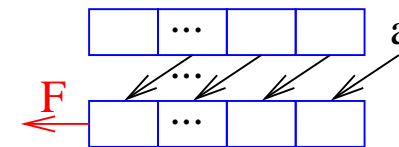


$$L = \{0, 1\}^* 1 \{0, 1\}^{k-1}$$

der Minimalautomat hat  $2^k$  Zustände.

$$(\{0, \dots, 2^k - 1\}, \{0, 1\}, \delta, 0, F)$$

$$\delta(q, a) = 2q + a, q \in F \Leftrightarrow q[k-1] = 1$$





## Nachweis von Nichtregularität

### □ Pumping Lemma:

- + : Leicht handhabbar
- : Nur notwendige Bedingung

### □ Nerode Relation

- + : Notwendige **und** hinreichende Bedingung  $\text{index}(R_L) = \infty$
- : Etwas unhandlicher



## Nerode Relation

**Beispiel:**  $L = \{a^n b^n : n \geq 1\}$

Behauptung:  $\forall k > 1, j \neq k > 1 : [a^k b] \neq [a^j b]$

$[a^k b] = \{a^k b, a^{k+1} b b, \dots\} = \{a^{k+i} b^{i+1}\}$

also immer  $k - 1$  mehr a's als b's.

Folglich sind  $[a^k b]$  und  $[a^j b]$  disjunkt

□



## Nerode Relation

**Beispiel:**  $L = \{c^m a^l b^l : m, l \geq 0\} \cup \{a, b\}^*$

Behauptung:  $\forall k > 1, j \neq k > 1 : [ca^k b] \neq [ca^j b]$

$[ca^k b] = \{c^m a^{k+i} b^{i+1} : m \geq 0, i \geq 1\}$

also immer  $k - 1$  mehr a's als b's.

Folglich sind  $[ca^k b]$  und  $[ca^j b]$  disjunkt

□



## 1.2.6 Abschlusseigenschaften

Seien  $L, L'$  reguläre Sprachen.

Dann sind folgende Sprachen ebenfalls regulär:

$L \cup L', L^*, L \cdot L'$ : nach Definition.

$\bar{L} := \Sigma^* \setminus L$ : Betrachte DFA  $A = (Q, \Sigma, \delta, s, F)$  mit  $L(A) = L$ .

Sei  $\bar{A} := (Q, \Sigma, \delta, s, Q \setminus F)$ . Dann gilt  $L(\bar{A}) = \bar{L}$ .

$L \cap L' = \overline{\bar{L} \cup \bar{L}'}$

(De Morgan)

$L \setminus L' = L \cap \bar{L}'$

$L^R$ : (Spiegelung) Übung. Hinweis: Induktion über reguläre Ausdrücke.



## Produktautomat

### Konstruktion eines DFA für Mengenoperationen

$L$  und  $L'$  seien reguläre Sprachen definiert durch DFAs

$$A = (Q, \Sigma, \delta, s, F),$$

$$A' = (Q', \Sigma, \delta', s', F').$$

Idee: Ein Automat  $A_{\times}$  emuliert das Verhalten von  $A$  und  $A'$ .

**Produktautomat:**  $A_{\times} := (Q \times Q', \Sigma, \delta_{\times}, (s, s'), F_{\times})$  mit  
 $\delta_{\times}((q, q'), a) = (\delta(q, a), \delta'(q', a))$

Definiere  $F$  je nach Mengenoperation:

$$L \cup L': F_{\times} := Q \times F' \cup F \times Q'$$

$$L \cap L': F_{\times} := F \times F'$$

...



## Leerheitsproblem

$L = \emptyset$ ?

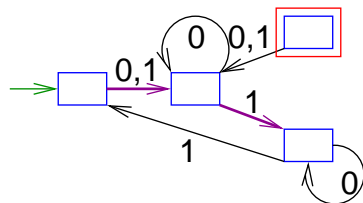
Representation DFA oder NFA  $A$ :

$L = \emptyset \Leftrightarrow \neg \exists f \in F : f$  ist von  $s$  aus **erreichbar**

$\rightsquigarrow$  Tiefensuche, **Linearzeit**, auch für **NFA**.

### Beispiel

$L(A) = \emptyset$



Kante im  
Tiefensuchbaum



## 1.2.7 Entscheidbarkeit

### einfacher Eigenschaften endlicher Automaten

### Wortproblem

$w \in L$ ?

Wandle in DFA  $A$  um.

Simuliere  $A$  bei Eingabe von  $w$ .

Endzustand erreicht?

Linearzeit falls DFA bekannt!



## Endlichkeitsproblem I — über Pumping Lemma

Sei  $n$  die Zahl aus dem Pumping-Lemma für  $L$  (Typ-3)

**Behauptung:**  $|L(G)| = \infty \Leftrightarrow \exists z \in L(G) : n \leq |z| < 2n$

**Beweis:**

$z \in L(G), n \leq |z| < 2n \rightarrow$  Pumpinglemma garantiert  $|L| = \infty$ .

Falls  $|L(G)| = \infty$  betrachte  $z \in L(G)$  mit minimalem  $|z| \geq n$ .

Annahme  $|z| \geq 2n$ .

$\xrightarrow{\text{Pumpinglemma}} z = uvw, 1 \leq |v| \leq |uv| \leq n, uv \in L(G) \rightarrow |uw| \geq n$ .

Widerspruch zur Minimalität von  $|z|$ .



### Endlichkeitsproblem II — Kreisdetektion

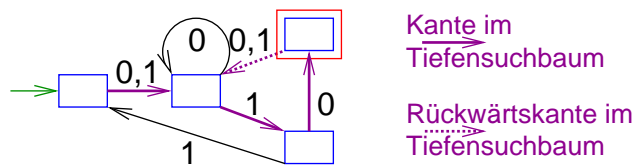
$|L(A)| = \infty? \Leftrightarrow \exists$  akzeptierender Pfad, der Kreis enthält.

OBdA: NFA mit  $F = \{f\}$ . Sei  $G_A = (Q, E)$ ,

$E = \{(q, r) : \exists a \in \Sigma \cup \{\epsilon\} : r \in \delta(q, a)\}$

1. Entferne Zustände, von denen aus  $f$  nicht erreichbar ist.  
Tiefensuche in  $\bar{G}_a = (Q, \{(q, r) : (r, q) \in E\})$  von  $f$ .

2. Ist von  $s$  aus ein Kreis erreichbar?  $\Leftrightarrow$   
Trifft Tiefensuche in  $G_A$  von  $s$  auf eine Rückwärtskante?



### Vollständigkeitsproblem

$L(A) = \Sigma^*$ ?

$\Leftrightarrow \neg \exists q \in Q \setminus F : q$  ist von  $s$  aus erreichbar?

$\rightsquigarrow$  Tiefensuche, **Linearzeit**, nur für **DFA!**

(äquivalent: Leerheit von  $\bar{L}$ )

Vollständigkeit für NFA:

Umwandlung in DFA. Nichts deutlich besseres bekannt.



### Äquivalenzproblem

$L$  und  $L'$  seien reguläre Sprachen definiert durch DFAs  $A, A'$ .

Frage  $L = L'?$

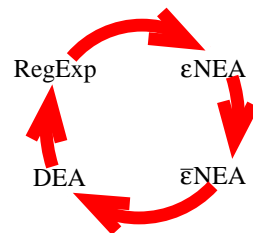
$\Leftrightarrow \neg \exists w : (w \in L \wedge w \notin L') \vee (w \notin L \wedge w \in L')$

$\Leftrightarrow \neg \exists w : (w \in L \wedge w \in \bar{L}') \vee (w \in \bar{L} \wedge w \in L')$

$\Leftrightarrow (L \cap \bar{L}') \cup (\bar{L} \cap L') = \emptyset$

z.B. via Produktautomat

Problem: langsam



### Äquivalenz von DFAs

$L$  und  $L'$  seien reguläre Sprachen definiert durch DFAs

$A = (Q, \Sigma, \delta, s, F), A' = (Q', \Sigma, \delta', s', F')$ .

Idee: der **Minimalautomat** ist „eindeutig“.

$\rightsquigarrow$  minimiere beide Automaten und prüfe auf „Gleichheit“.

Problem: Zustandsumbenennungen sind erlaubt. Die Komplexität des

**Isomorphietestes** allgemeiner Graphen ist ein offenes Problem.



## Äquivalenz von DFAs

$L$  und  $L'$  seien reguläre Sprachen definiert durch DFAs

$A = (Q, \Sigma, \delta, s, F), A' = (Q', \Sigma, \delta', s', F')$ . ObdA  $Q \cap Q' = \emptyset$ .

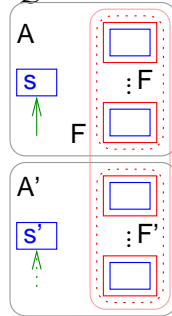
Frage  $L = L'$ ?

Betrachte  $A_{\cup} := (Q \cup Q', \Sigma, \delta_{\cup}, s, F \cup F')$  mit

$$\delta_{\cup}(q, a) = \begin{cases} \delta(q, a) & \text{falls } q \in Q \\ \delta'(q, a) & \text{falls } q \in Q' \end{cases}$$

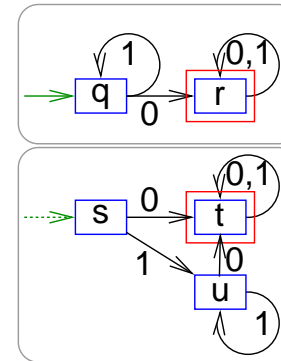
Bestimme Äquivalenzklassen der Zustände von  $A_{\cup}$ .

$L = L' \Leftrightarrow s \equiv s'$ .



## Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit mindestens einer Null



Algorithmus zur Markierung aller

inäquivalenter Zustandspaare liefert:

$\{q, r\}, \{q, t\}, \{s, r\}, \{s, t\}, \{u, r\}, \{u, t\}$

$\rightsquigarrow q \equiv s$

$\rightsquigarrow$  Beide Automaten sind äquivalent.



## Zusammenfassung Endliche Automaten u. Reguläre Sprachen

- Einfaches Maschinenmodell
- Umfassend algorithmisch beherrschbar (Zustandsminimierung, Konvertierung mit regulären Ausdrücken, ...)
- Akzeptierte Sprachen vollständig verstanden
- Nützliche Anwendungen: Textverarbeitung, Compiler, Hardware, ...
- Konzept des Nichtdeterminismus