

# Endliche Automaten

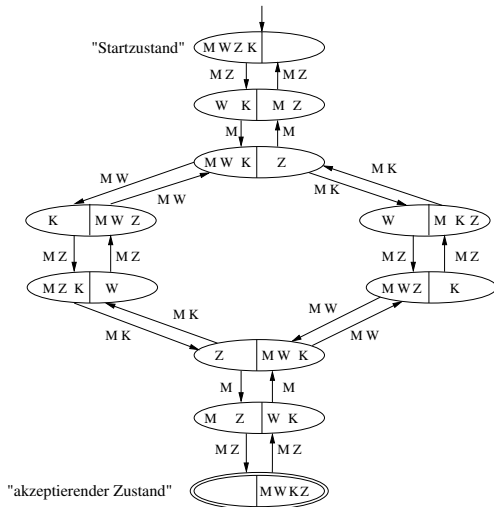
Endliche Automaten erlauben eine **Beschreibung von Handlungsabläufen**:

Wie ändert sich ein Systemzustand in Abhängigkeit von veränderten Umgebungsbedingungen?

Vielfältiges Einsatzgebiet, nämlich:

- in der Definition der **regulären Sprachen**  
also der Menge aller Folgen von Ereignissen,  
die von einem Startzustand in einen gewünschten Zustand führen,
- in der **Entwicklung digitaler Schaltungen**
- in der **Softwaretechnik** (z. B. in der Modellierung des Applikationsverhaltens)
- in der **Compilierung**: Lexikalische Analyse
- im **Algorithmenentwurf** für String Probleme
- in der **Abstraktion tatsächlicher Automaten** (wie Bank- und Getränkeautomaten, Fahrstühle etc.).

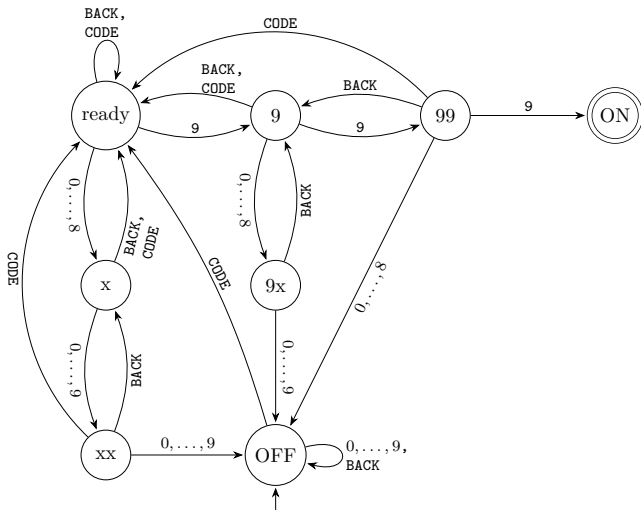
# Transitionssysteme: Das Problem der Flußüberquerung



Dieser endliche Automat „akzeptiert“ genau diejenigen Folgen von einzelnen Flussüberquerungen, die vom Startzustand in den akzeptierenden Zustand führen.

- Um die Kindersicherung des Fernsehers über die Fernbedienung freizuschalten, muss ein dreistelliger Code korrekt eingegeben werden. Dabei sind die folgenden Tasten relevant:
  - Die Tasten  $0, \dots, 9$ ,
  - die Taste CODE sowie
  - die Taste BACK.
- Wird BACK gedrückt, so wird die zuletzt eingegebene Zahl zurückgenommen.
- Die Taste CODE muss vor Eingabe des Codes gedrückt werden. Wird CODE während der Codeeingabe nochmals gedrückt, so wird die Eingabe neu begonnen.

Der Code zum Entsperren ist 999.



Der Automat „**akzeptiert**“ alle Folgen von Bedienoperationen, die vom Zustand „**OFF**“ in den Zustand „**ON**“ führen.

# Alphabete, Worte und Sprachen

# Worte und Alphabete

Sei  $\Sigma$  eine endliche Menge: Wir nennen  $\Sigma$  ein **Alphabet**.

- Wir fassen ein Tupel

$$w = (a_1, \dots, a_k) \in \Sigma^k$$

als ein **Wort** mit den „Buchstaben“  $a_1, \dots, a_k$  auf und schreiben oft

$$w = a_1 \cdots a_k.$$

- ▶ Das leere Tupel  $() \in \Sigma^0$  heißt auch das **leere Wort** und wird oft mit  $\varepsilon$  (epsilon) bezeichnet.
- Die **Länge**  $|a_1 \cdots a_k|$  eines Wortes  $a_1 \cdots a_k$  ist die Zahl  $k$ , die Anzahl seiner Buchstaben.
  - ▶ Insbesondere ist  $|\varepsilon| = 0$ , das leere Wort hat also die Länge 0.
- Sind  $v = a_1 \cdots a_k$  und  $w = b_1 \cdots b_\ell$  zwei Worte über  $\Sigma$ , so ist die **Konkatenation** von  $v$  und  $w$  das Wort

$$vw := a_1 \cdots a_k b_1 \cdots b_\ell.$$

Sei  $\Sigma$  ein Alphabet. (Also ist  $\Sigma$  eine Menge.)

(a) Die **Menge aller Worte über  $\Sigma$**  bezeichnen wir mit  $\Sigma^*$ . Es gilt also:

$$\Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k = \{ a_1 \cdots a_k : k \in \mathbb{N}, a_1, \dots, a_k \in \Sigma \}.$$

Es ist  $\Sigma^0 = \{()\} = \{\varepsilon\}$  und  $\Sigma^*$  enthält insbesondere das leere Wort.

(b) Die Menge aller **nicht-leeren** Worte über  $\Sigma$  bezeichnen wir mit  $\Sigma^+$ . Es gilt:

$$\Sigma^+ = \Sigma^* \setminus \{\varepsilon\} = \{ a_1 \cdots a_k : k \in \mathbb{N}_{>0}, a_1, \dots, a_k \in \Sigma \}.$$

(c) Eine Teilmenge von  $\Sigma^*$ , also eine *Menge von Worten* über  $\Sigma$ , wird eine **Sprache** über  $\Sigma$  genannt.

**Bemerkung:** In vielen Büchern werden Sprachen mit dem Buchstaben  $L$  (für **L**anguage) oder mit Varianten wie  $L'$  oder  $L_1$  bezeichnet.



Wir betrachten das Alphabet

$$\Sigma_{\text{deutsch}} := \{A, B, \dots, Z, \ddot{A}, \ddot{O}, \ddot{U}, a, b, \dots, z, \ddot{a}, \ddot{o}, \ddot{u}, \beta, \cdot, ,, :, ;, !, ?, -, \_ \}.$$

Beispiele für Sprachen über  $\Sigma_{\text{deutsch}}$  sind:

- $L_1 :=$  Menge aller **Worte** der deutschen Sprache.
- $L_2 :=$  Menge aller **grammatikalisch korrekten Sätze** der deutschen Sprache aufgefasst als Worte über  $\Sigma_{\text{deutsch}}$ ,
  - ▶ Die Menge aller in Deutsch geschriebenen **Bücher**, die nur aus grammatikalisch korrekten Sätzen aufgebaut sind, ist eine Teilmenge von

$$L_2^+.$$

Wir betrachten das Alphabet ASCII (american standard code for information interchange)

ASCII := die Menge aller ASCII-Symbole.

ASCII besteht aus 95 druckbaren Zeichen (das lateinische Alphabet in Groß- und Kleinschreibung, die arabischen Ziffern, Sonderzeichen) und 33 nicht druckbaren Zeichen. Beispiele für Sprachen über dem Alphabet ASCII sind:

- $L_1$  := die Menge aller Python-Schlüsselwörter,
- $L_2$  := die Menge aller erlaubten Variablennamen in Python,
- $L_3$  := die Menge aller syntaktisch korrekten Python-Programme.

$L_1, L_2, L_3 \subseteq \text{ASCII}^*$ .

Im Folgenden interessieren wir uns vor Allem für Sprachen, die von endlichen Automaten definiert werden.

Im „Fernseher-Beispiel“ interessieren wir uns etwa für die Sprache aller Worte über dem Alphabet

$$\Sigma := \{0, 1, \dots, 9\} \cup \{\text{CODE}, \text{BACK}\},$$

die zu einer Freischaltung des Fernsehers führen.

## DFAs: Die formale Definition

# Was ist ein endlicher Automat?

Ein **deterministischer endlicher Automat** (engl. deterministic finite automaton, kurz **DFA**)

$$A = (\Sigma, Q, \delta, q_0, F)$$

besteht aus:

- einer endlichen Menge  $\Sigma$ , dem **Eingabealphabet**,
- einer endlichen Menge  $Q$ , der **Zustandsmenge** (die Elemente aus  $Q$  werden Zustände genannt),
- einer Funktion  $\delta$  von  $Q \times \Sigma$  nach  $Q$ , der **Übergangsfunktion** (oder Überföhrungsfunktion),
- einem Zustand  $q_0 \in Q$ , dem **Startzustand**,
- sowie einer Menge  $F \subseteq Q$  von **Endzuständen** bzw. **akzeptierenden Zuständen**. (Der Buchstabe  $F$  steht für „final states“, also „Endzustände“).

Das Zustandsdiagramm (bzw. der Automatengraph),  
die grafische Darstellung eines DFA

# Grafische Darstellung: Das Zustandsdiagramm

$A = (\Sigma, Q, \delta, q_0, F)$  sei ein DFA.

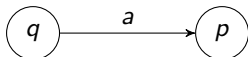
- Für jeden Zustand  $q \in Q$  gibt es einen durch  $\circlearrowleft q$  dargestellten Knoten.
- Der Startzustand  $q_0$  wird durch einen in ihn hinein führenden Pfeil markiert, d.h.:



- Jeder akzeptierende Zustand  $q \in F$  wird durch eine doppelte Umrandung markiert, d.h.:

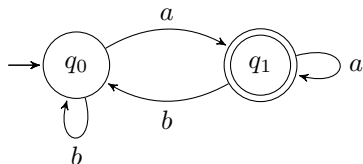


- Seien  $p, q \in Q$  Zustände und  $a \in \Sigma$  ein Symbol des Alphabets mit  $\delta(q, a) = p$ . Dann füge einen mit dem Symbol  $a$  beschrifteten Pfeil von Knoten  $\circlearrowleft q$  zu Knoten  $\circlearrowleft p$  ein, d.h.:



# Vom Zustandsdiagramm zur formalen Beschreibung

Der DFA  $A_1$  hat das Zustandsdiagramm



Wir möchten  $A_1$  formal beschreiben.

- 1 Das Eingabealphabet  $\Sigma = \{a, b\}$ ,
- 2 die Zustandsmenge  $Q = \{q_0, q_1\}$
- 3 die Übergangsfunktion  $\delta : Q \times \Sigma \rightarrow Q$ : siehe Tafel.
- 4 der Startzustand  $q_0$  und
- 5 die Menge  $F = \{q_1\}$  der akzeptierenden Zustände.



# Wie arbeitet ein DFA?

## Die erweiterte Übergangsfunktion

Ein DFA  $A = (\Sigma, Q, \delta, q_0, F)$  erhält als Eingabe ein Wort  $w \in \Sigma^*$ .  
(Das Wort repräsentiert eine Folge von „Aktionen“ oder „Bedienoperationen“.)

$A$  wird im Startzustand  $q_0$  gestartet.

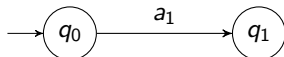
0. Falls  $w$  das leere Wort ist, d.h.  $w = \varepsilon$ , dann verbleibt  $A$  im Zustand  $q_0$ .
1. Also gelte  $w = a_1 \cdots a_n$  mit  $n \in \mathbb{N}_{>0}$  und  $a_1, \dots, a_n \in \Sigma$ . Der Automat liest den ersten Buchstaben  $a_1$  von  $w$  und wechselt in den Zustand

$$q_1 := \delta(q_0, a_1).$$

In der grafischen Darstellung von  $A$  wird der Zustand



durch die mit  $a_1$  beschriftete Kante verlassen, und  $q_1$  ist der Endknoten dieser Kante, d.h.



2. Durch Lesen von  $a_2$ , dem zweiten Symbol von  $w$ , wechselt  $A$  in den Zustand

$$q_2 := \delta(q_1, a_2).$$

In der grafischen Darstellung von  $A$  wird  $(q_1)$  durch die mit  $a_2$  beschriftete

Kante  $(q_1) \xrightarrow{a_2} (q_2)$  verlassen und der Automat landet in Zustand  $q_2 = \delta(q_1, a_2)$ .

- $n$ . Auf diese Weise wird das gesamte Eingabewort  $w = a_1 \cdots a_n$  abgearbeitet.
- ▶ Ausgehend vom Startzustand  $q_0$  erreicht  $A$  nacheinander Zustände  $q_1, \dots, q_n$ .
  - ▶ In der grafischen Darstellung von  $A$  entspricht diese Zustandsfolge einem Weg der Länge  $n$ , der im Knoten



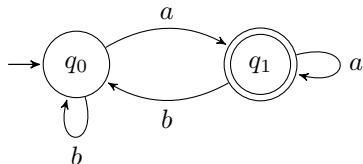
startet und dessen Kanten mit den Buchstaben  $a_1, \dots, a_n$  beschriftet sind.

- ▶ Schreibe

$$\widehat{\delta}(q_0, w) := q_n.$$

# Die erweiterte Übergangsfunktion für den DFA $A_1$

Wir betrachten wieder den DFA  $A_1$  mit dem Zustandsdiagramm



- 1  $\widehat{\delta}(q_0, ab) = q_0$
- 2  $\widehat{\delta}(q_0, ababab) = q_0$
- 3  $\widehat{\delta}(q_1, ababab) = q_0$
- 4  $\widehat{\delta}(q_1, abbaba) = q_1$
- 5  $\widehat{\delta}(q_0, (ba)^{100}) = q_1$

# Was genau ist $\hat{\delta}(q, w)$ ?

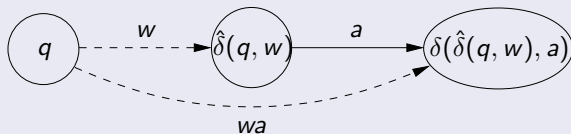
Sei  $A := (\Sigma, Q, \delta, q_0, F)$  ein DFA. Eine rekursive Definition der Funktion

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

- Der **Rekursionsanfang**: f.a.  $q \in Q$  ist  $\hat{\delta}(q, \varepsilon) := q$ .
- Der **Rekursionsschritt**: F.a.  $q \in Q$ ,  $w \in \Sigma^*$  und  $a \in \Sigma$  gilt für  $q' := \hat{\delta}(q, w)$ :

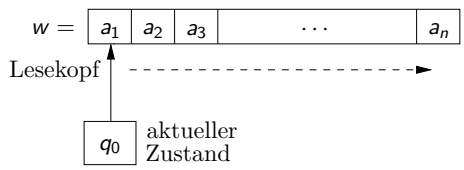
$$\hat{\delta}(q, wa) := \delta(q', a).$$

Grafische Darstellung:



$\hat{\delta}(q_0, w)$  ist der Zustand, der nach Verarbeitung des Worts  $w$  erreicht wird.

Verarbeitung eines Eingabeworts  
durch einen DFA  $A$ :



Wir können uns einen DFA als eine Maschine vorstellen, die

- \* ihre Eingabe  $w = a_1 a_2 a_3 \dots a_n$   
von links-nach-rechts mit Hilfe eines Lesekopfes durchläuft
- \* und dabei Zustandsübergänge durchführt.

## Die akzeptierte Sprache eines DFA

# Wann akzeptiert $A = (\Sigma, Q, \delta, q_0, F)$ ein Wort $w$ ?

**DEFINITION:** Das Eingabewort  $w$  wird vom DFA  $A$  **akzeptiert**, falls

$$\widehat{\delta}(q_0, w) \in F,$$

d.h., falls der nach Verarbeitung von  $w$  erreichte Zustand zur Menge  $F$  der akzeptierenden Zustände gehört.


Wie „übersetzt“ sich

$$\text{Akzeptanz von } w = a_1 \cdots a_n$$

in der grafischen Darstellung von  $A$ ? Es gibt einen in



startenden Weg der Länge  $n$ ,

- dessen Kanten mit den Symbolen  $a_1, \dots, a_n$  beschriftet sind,
- und der in einem akzeptierenden Zustand  endet.



# Die akzeptierte Sprache $L(A)$

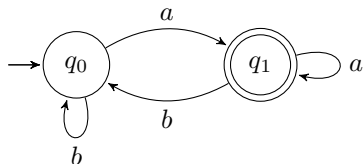
Die von einem DFA  $A = (\Sigma, Q, \delta, q_0, F)$  akzeptierte Sprache  $L(A)$  ist

$$L(A) := \{ w \in \Sigma^* : \hat{\delta}(q_0, w) \in F \}.$$

*Ein Wort  $w \in \Sigma^*$  gehört also genau dann zur Sprache  $L(A)$ , wenn  $w$  vom DFA  $A$  akzeptiert wird.*

# Der Automat $A_1$

Wir betrachten wieder den DFA  $A_1$  mit dem Zustandsdiagramm

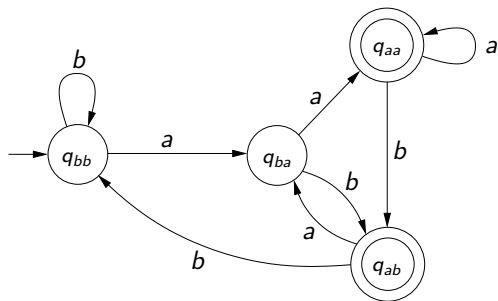


$A_1$  akzeptiert die Sprache

$$L(A_1) = \{w \in \{a, b\}^* : \text{der letzte Buchstabe von } w \text{ ist ein } a \}$$

# Der Automat $A_2$

Der DFA  $A_2$  mit dem Zustandsdiagramm



akzeptiert die Sprache

$$L(A_2) = \{w \in \{a, b\}^* : \text{der vorletzte Buchstabe von } w \text{ ist ein } a \}$$

# Ein Paritätscheck

Bei der Speicherung von Daten auf einem Speichermedium eines Computers werden Informationen durch binäre Worte über dem Alphabet  $\{0, 1\}$  **kodiert**.

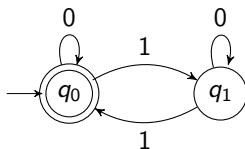
Um Fehler bei der Datenübertragung zu erkennen, wird oft ein „**Paritätsbit**“ angehängt, so dass die Summe der Einsen im resultierenden Wort  $w$  gerade ist.

Für ein beliebiges Wort  $w \in \{0, 1\}^*$  sagen wir

$w$  „besteht den Paritätscheck“,

falls die Anzahl der Einsen in  $w$  **gerade** ist.

Der folgende DFA  $A$  führt einen Paritätscheck durch:



Für  $A$  gilt:  $L(A) = \{w \in \{0, 1\}^* : w \text{ besteht den Paritätscheck}\}$ .

# Minimierung von DFAs

# Minimiere die Zustandszahl

$A = (\Sigma, Q^A, \delta^A, q_0^A, F^A)$  und  $B = (\Sigma, Q^B, \delta^B, q_0^B, F^B)$  seien DFAs.

(a) Wir nennen  $A$  und  $B$  **äquivalent**, wenn gilt

$$L(A) = L(B).$$

(b)  $A$  heißt

**minimal**,

wenn kein mit  $A$  äquivalenter vollständiger DFA eine **kleinere** Zustandszahl besitzt.

## DAS ZIEL:

- Gegeben ist ein DFA  $A$ .
- Bestimme einen minimalen, mit  $A$  äquivalenten DFA.

# Zustandsminimierung: Die Idee

Der DFA  $A = (\Sigma, Q, \delta, q_0, F)$  sei gegeben.

**Die Idee:** Wir sollten doch zwei Zustände  $p, q \in Q$  zu einem einzigen Zustand verschmelzen dürfen, wenn  $p$  und  $q$  „**äquivalent**“ sind,

also dasselbe Ausgabeverhalten besitzen.

Aber was bedeutet das?

Die **Verschmelzungsrelation**  $\equiv_A$  ist eine 2-stellige Relation über der Zustandsmenge  $Q$ . Wir sagen, dass Zustände  $p, q \in Q$  **äquivalent bzgl. A** sind, (kurz  $p \equiv_A q$ ), wenn

$$\text{f.a. Worte } w \in \Sigma^* : \left( \widehat{\delta}(p, w) \in F \Leftrightarrow \widehat{\delta}(q, w) \in F \right).$$

Wir nennen  $\equiv_A$  **Verschmelzungsrelation**, da wir bzgl.  $\equiv_A$  äquivalente Zustände in einen Zustand verschmelzen möchten.

## Einschub: Äquivalenzrelationen



# 2-stellige Relationen und gerichtete Graphen

Zur Erinnerung: Für eine 2-stellige Relation  $R$  über der Knotenmenge  $V$  gilt

$$R \subseteq V \times V.$$

Wir können somit

- die Relation  $R$  als Kantenmenge und
- umgekehrt eine Kantenmenge als eine 2-stellige Relation auffassen.

Gerichtete Graphen mit Knotenmenge  $V$

und

2-stellige Relationen über  $V$

sind äquivalente Konzepte!

# Wichtige Eigenschaften 2-stelliger Relationen

Sei  $E$  eine 2-stellige Relation über einer Menge  $V$ , d.h.  $G = (V, E)$  ist ein gerichteter Graph.

(a)  $E$  heißt **reflexiv**, falls für alle  $v \in V$  gilt:

$$(v, v) \in E. \quad (\text{Skizze: } v \begin{array}{c} \circlearrowleft \end{array})$$

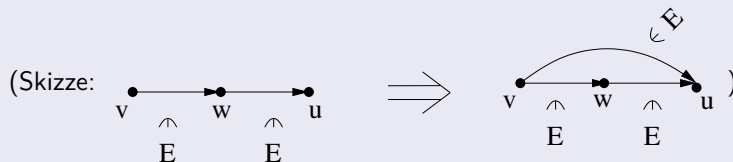
(b)  $E$  heißt **symmetrisch**, falls f.a.  $v, w \in V$  gilt:

Wenn  $(v, w) \in E$ , dann auch  $(w, v) \in E$ .

Zur Kante  $v \xrightarrow{\quad} w$  gibt es auch die "Rückwärtskante"  $w \xleftarrow{\quad} v$

(c)  $E$  heißt **transitiv**, falls f.a.  $v, w, u \in V$  gilt:

Ist  $(v, w) \in E$  und  $(w, u) \in E$ , so auch  $(v, u) \in E$ .



- (a) Eine **Äquivalenzrelation** ist eine 2-stellige Relation, die **reflexiv**, **symmetrisch** und **transitiv** ist.
- (b) Sei  $E$  eine Äquivalenzrelation über einer Menge  $V$ .
- ▶ Für jedes  $v \in V$  bezeichnet

$$[v]_E := \{v' \in V \mid (v, v') \in E\}$$

die **Äquivalenzklasse von  $v$**  bezüglich  $E$ .

- ★  $[v]_E$  besteht aus allen Elementen von  $V$ , die gemäß  $E$  "äquivalent" zu  $v$  sind.
- ▶ Eine Menge  $W \subseteq V$  heißt **Äquivalenzklasse** (bzgl.  $E$ ), falls  $W = [v]_E$  für ein Element  $v \in V$  gilt.
- ★ Das Element  $v$  heißt **Vertreter** (oder **Repräsentant**) seiner Äquivalenzklasse  $W$ .

# Die wichtigste Eigenschaft von Äquivalenzrelationen

Sei  $E$  eine Äquivalenzrelation über der Menge  $V$ .

(a) Dann gilt für alle Elemente  $v, w \in V$

$$[v]_E = [w]_E \text{ oder } [v]_E \cap [w]_E = \emptyset.$$

(b)  $V$  ist eine disjunkte Vereinigung von Äquivalenzklassen.

Beweis: Siehe Tafel.

(a) **Gleichheit:** Für jede Menge  $M$  ist

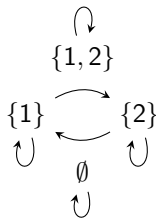
$$E := \{(m, m) : m \in M\}$$

eine Äquivalenzrelation: „ $(x, y) \in E$ “ gilt genau dann, wenn „ $x = y$ “.

(b) **Gleichmächtigkeit:** Für jede endliche Menge  $M$  ist

$$E := \{(A, B) : A \subseteq M, B \subseteq M, |A| = |B|\}$$

eine Äquivalenzrelation über der Potenzmenge  $\mathcal{M}$ . Skizze für  $M = \{1, 2\}$ :



(c) **Logische Äquivalenz:** Die Relation

$$E := \{(\phi, \psi) : \phi, \psi \in \text{AL}, \phi \equiv \psi\}$$

ist eine Äquivalenzrelation über der Menge AL aller aussagenlogischen Formeln.

(d) **Graph-Isomorphie:** Für jede Menge  $V$  ist

$$E := \{(G_1, G_2) \mid G_1 = (V, E_1), G_2 = (V, E_2) \\ \text{sind isomorphe, ungerichtete Graphen}\}$$

eine Äquivalenzrelation über der Menge aller ungerichteten Graphen mit Knotenmenge  $V$ .

Sei  $E$  eine Äquivalenzrelation. Dann ist der **Index** von  $E$  die Anzahl der verschiedenen Äquivalenzklassen.

*Der Index einer Äquivalenzrelation gibt an, wie viele verschiedene Äquivalenzklassen es gibt.*

Betrachte wieder die Äquivalenzrelation der Gleichmächtigkeit für Teilmengen  $A, B \subseteq M$ , also

$$A \equiv B \Leftrightarrow |A| = |B|.$$

Dann stimmt der Index überein mit  $|M| + 1$ .

Zurück zur Verschmelzungsrelation  $\equiv_A$



# Die Verschmelzungsrelation ist eine Äquivalenzrelation

Der DFA  $A = (\Sigma, Q, \delta, q_0, F)$  sei gegeben. Zur Erinnerung:  $p \equiv_A q \iff$

f.a. Worte  $w \in \Sigma^*$  gilt:  $(\widehat{\delta}(p, w) \in F \iff \widehat{\delta}(q, w) \in F)$ .

(a) Die Verschmelzungsrelation ist

- ▶ **reflexiv**, f.a.  $p \in Q$ :  $p \equiv_A p$ ,
- ▶ **symmetrisch**, f.a.  $p, q \in Q$ : wenn  $p \equiv_A q$ , dann  $q \equiv_A p$  und
- ▶ **transitiv**, f.a.  $p, q, r \in Q$ : wenn  $p \equiv_A q$  und  $q \equiv_A r$ , dann  $p \equiv_A r$ .

Warum? Siehe Tafel.

(b) Die Verschmelzungsrelation ist eine Äquivalenzrelation!

Die Zustandsmenge  $Q$  ist die disjunkte Vereinigung der Äquivalenzklassen von  $\equiv_A$ .

# Die nächsten Schritte

Sei der DFA  $A = (\Sigma, Q, \delta, q_0, F)$  gegeben.

Wir führen die folgenden Schritte durch:

1. Wir bestimmen die Äquivalenzklassen der Verschmelzungsrelation  $\equiv_A$ .
2. Für jede Äquivalenzklasse von  $\equiv_A$  verschmelzen wir alle Zustände der Klasse zu einem einzigen Zustand und fügen „entsprechende“ Übergänge ein.

*Und wie genau soll das aussehen?*

3. Den neuen Automaten nennen wir  $A'$  und bezeichnen ihn als den **Äquivalenzklassenautomaten** von  $A$ .

Die Anzahl der Zustände von  $A'$  stimmt mit dem Index von  $\equiv_A$  überein.

**Wenn** der Index mit der minimalen Zustandszahl übereinstimmt, dann ist  $A'$  **minimal!**

Schritt 1:  
Wir bestimmen die Verschmelzungsrelation  $\equiv_A$

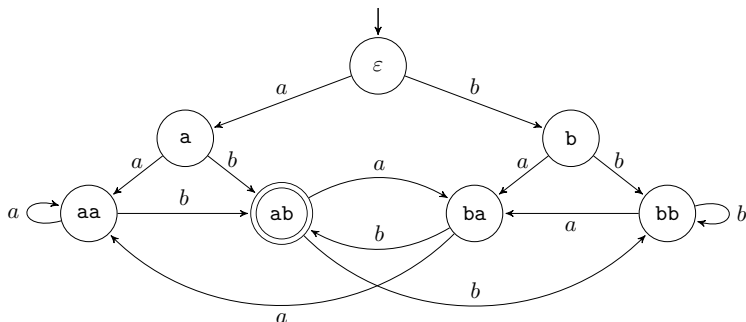
Sei  $(\Sigma, Q, \delta, q_0, F)$  ein DFA.

(a) Das Wort  $w \in \Sigma^*$  ist **Zeuge** für die Inäquivalenz von  $p$  und  $q$ , wenn

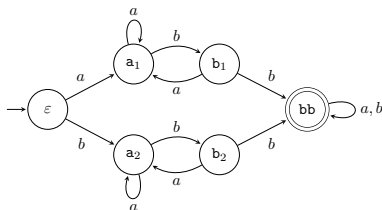
$$\left( \widehat{\delta}(p, w) \in F \wedge \widehat{\delta}(q, w) \notin F \right) \vee \left( \widehat{\delta}(p, w) \notin F \wedge \widehat{\delta}(q, w) \in F \right).$$

Wir sagen auch, dass  $w$  die Zustände  $p$  und  $q$  **trennt**.

(b) Es ist  $p \not\equiv_A q$  genau dann, wenn es einen Zeugen für die Inäquivalenz von  $p$  und  $q$  gibt.



- Finde einen Zeugen für die Inäquivalenz von  $(ab)$  und  $(ba)$ .
- Welcher Zeuge trennt  $(\epsilon)$  und  $(a)$  ?
- Gibt es Zeugen, die die Zustände  $(a)$  und  $(aa)$  trennen?



- Finde einen Zeugen für die Inäquivalenz von  $b_1$  und  $bb$ .
- Welcher Zeuge trennt  $\varepsilon$  und  $a_1$  ?
- Gibt es Zeugen, die die Zustände  $a_1$  und  $a_2$  trennen?

# Wir bestimmen alle Paare **nicht-äquivalenter** Zustände

1. Füge alle Paarmengen  $\{p, q\}$  mit  $p \in F$  und  $q \notin F$  in die Menge  $M_0$  ein.
  - ▶ Es ist  $\delta(p, \varepsilon) \in F$  und  $\delta(q, \varepsilon) \notin F$ .
  - ▶  $w = \varepsilon$  ist **Zeuge** für die **Nicht-Äquivalenz** von  $p$  und  $q$ .
2. Die Paarmenge  $\{p, q\}$  gehöre nicht zu  $M_i$ , aber  $\{r, s\}$  sei Element von  $M_i$ .

$$\text{Wenn } \delta(p, a) = r \text{ sowie } \delta(q, a) = s$$

für ein  $a \in \Sigma$ , dann füge  $\{p, q\}$  in die Menge  $M_{i+1}$  ein.

- ▶ Da  $r \neq_A s$ , gibt es einen **Zeugen**  $w$  mit

$$(\widehat{\delta}(r, w) \in F \text{ und } \widehat{\delta}(s, w) \notin F) \text{ oder } (\widehat{\delta}(r, w) \notin F \text{ und } \widehat{\delta}(s, w) \in F).$$

- ▶ Das Wort  $aw$  **trennt**  $p$  und  $q \implies aw$  ist **Zeuge** für die **Nicht-Äquivalenz** von  $p$  und  $q$ .

3. Halte, wenn keine neuen Paarmengen  $\{p, q\}$  als nicht-äquivalent nachgewiesen werden können.
  - ▶ Wir behaupten, dass  $p \neq_A q$  genau dann gilt, wenn die Paarmenge  $\{p, q\}$  zu einer Menge  $M_i$  gehört.

Stimmt die Behauptung: Finden wir alle Paare nicht-äquivalenter Zustände?

# Unser Verfahren funktioniert!

Sei  $P$  die Menge aller Paare  $\{p, q\}$  nicht-äquivalenter Zustände, die aber von unserem Verfahren **nicht** gefunden werden. **Zeige**, dass  $P$  leer ist!

Angenommen,  $P$  ist nicht-leer. Die Paarmenge  $\{p, q\} \in P$  habe unter allen Paarmengen in  $P$  einen **kürzesten** Zeugen  $w$ .

1. Wenn  $w = \varepsilon$ , dann ist

- ▶  $(\delta(p, \varepsilon) \in F \text{ und } \delta(q, \varepsilon) \notin F)$  oder  $(\delta(p, \varepsilon) \notin F \text{ und } \delta(q, \varepsilon) \in F)$ ,
- ▶ bzw.  $(p \in F \text{ und } q \notin F)$  oder  $(p \notin F \text{ und } q \in F)$ .

Aber dann haben wir  $\{p, q\}$  in die Menge  $M_0$  eingefügt. ⚡

2. Wenn  $w = au$  für den Buchstaben  $a \in \Sigma$ , dann ist

- ▶  $(\widehat{\delta}(p, au) \in F \text{ und } \widehat{\delta}(q, au) \notin F)$  oder  $(\widehat{\delta}(p, au) \notin F \text{ und } \widehat{\delta}(q, au) \in F)$ ,
- ▶ bzw.  $(\widehat{\delta}(\delta(p, a), u) \in F \text{ und } \widehat{\delta}(\delta(q, a), u) \notin F)$  oder  $(\widehat{\delta}(\delta(p, a), u) \notin F \text{ und } \widehat{\delta}(\delta(q, a), u) \in F)$ .

Aber dann ist  $\delta(p, a) \not\equiv_A \delta(q, a)$  mit dem **kürzeren** Zeugen  $u$ :

- ▶ Nach Annahme haben wir  $\{\delta(p, a), \delta(q, a)\}$  in eine Menge  $M_i$  eingefügt
- ▶ und werden darauffolgend  $\{p, q\}$  in die Menge  $M_{i+1}$  einfügen. ⚡



# Der Äquivalenzklassenautomat

# Der Äquivalenzklassenautomat

Wie sieht der Automat nach dem Verschmelzen aller äquivalenten Zustände aus?

- Für Zustand  $p \in Q$  bezeichnet

$$[p]_A := \{q \in Q : p \equiv_A q\}$$

die Äquivalenzklasse von  $p$ .

- Der **Äquivalenzklassenautomat**  $A'$  für  $A$  besitzt

- ▶ die Zustandsmenge

$$Q' := \{ [p]_A : p \in Q \},$$

- ▶ den Anfangszustand  $q'_0 := [q_0]_A$ ,
- ▶ die Menge  $F' := \{ [p]_A : p \in F \}$  der akzeptierenden Zustände und
- ▶ das Programm  $\delta'$  mit

$$\delta'([p]_A, a) := [ \delta(p, a) ]_A$$

für alle Zustände  $q \in Q$  und Buchstaben  $a \in \Sigma$ .

# Der Minimierungsalgorithmus

# Berechnung von $\equiv_A$ und $A'$

**Eingabe:** Ein DFA  $A = (Q, \Sigma, \delta, q_0, F)$ .

**Schritt 1:** Entferne aus  $A$  alle **überflüssigen** Zustände, d.h. alle Zustände, die nicht von  $q_0$  aus erreichbar sind.

**Schritt 2:** Bestimme alle Paarmengen  $\{p, q\}$  mit  $p, q \in Q$  und  $p \not\equiv_A q$ :

(a) Zuerst bestimme  $M_0 := \{ \{p, q\} : p \in F, q \in Q \setminus F \}$ ; Setze  $i := 0$

(b) Wiederhole

(c) Für alle Paarmengen  $\{p, q\}$  mit  $p \neq q$ ,  $\{p, q\} \notin (M_0 \cup \dots \cup M_i)$  und für alle  $a \in \Sigma$  tue folgendes:

(d) Falls  $\{\delta(p, a), \delta(q, a)\} \in M_i$ , füge  $\{p, q\}$  zur Menge  $M_{i+1}$  hinzu.

(e)  $i := i + 1$

(f) bis  $M_i = \emptyset$

(g) **Ausgabe:**  $M := M_0 \cup \dots \cup M_{i-1}$ .

**Schritt 3:** Konstruiere  $A' := (Q', \Sigma, \delta', q'_0, F')$ :

$Q' := \{ [q]_A : q \in Q \}$ , wobei  $[q]_A = \{ p \in Q : \{p, q\} \notin M \}$

$q'_0 := [q_0]_A$ ,  $F' := \{ [q]_A : q \in F \}$

$\delta' : Q' \times \Sigma \rightarrow Q'$  mit  $\delta'([q]_A, a) := [\delta(q, a)]_A$  für alle  $q \in Q$  und  $a \in \Sigma$ .

Wir definieren eine Äquivalenzrelation  $\equiv_A^i$  auf der Zustandsmenge  $Q$  des Automaten  $A = (\Sigma, Q, \delta, q_0, F)$  durch

$$p \equiv_A^i q \quad :\Leftrightarrow \quad \text{für alle Wörter } w \in \Sigma^0 \cup \Sigma^1 \cup \dots \cup \Sigma^i \text{ gilt} \\ \left( \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F \right)$$

Zwei Zustände  $p$  und  $q$  sind also genau dann für  $\equiv_A^i$  äquivalent, wenn sie von keinem Zeugen der Länge **höchstens**  $i$  voneinander getrennt werden können.

$$p \equiv_A^i q \iff \text{für alle Wörter } w \in \Sigma^0 \cup \Sigma^1 \cup \dots \cup \Sigma^i \text{ gilt} \\ \left( \hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F \right)$$

Zeige mit vollständiger Induktion nach  $i$ :

$$p \equiv_A^i q \iff \{p, q\} \notin M_0 \cup \dots \cup M_i.$$

1. Es ist  $p \equiv_A^0 q \iff \left( (p \in F \text{ und } q \in F) \text{ oder } (p \notin F \text{ und } q \notin F) \right)$ . Also folgt

$$\{p, q\} \notin M_0 \iff p \equiv_A^0 q$$

2.

$$p \equiv_A^{i+1} q \iff \underbrace{(p \equiv_A^i q)}_{\text{wende Ind.Vor. an}} \text{ und für alle } a \in \Sigma \text{ gilt: } \underbrace{(\delta(p, a) \equiv_A^i \delta(q, a))}_{\text{wende Ind.Vor. an}}.$$

Der Minimierungsalgorithmus versucht, aus den Äquivalenzklassen von  $\equiv_A^i$  einen DFA zu bauen. Dieser Versuch scheitert genau dann, wenn vermeintlich äquivalente Zustände keine äquivalenten Nachfolger besitzen, d.h. wenn gilt

$$\delta(p, a) \not\equiv_A^i \delta(q, a) \text{ für einen Buchstaben } a \in \Sigma \text{ und Zustände } p, q \text{ mit } p \equiv_A^i q.$$

Aber dann wird  $\{p, q\}$  zur Menge  $M_{i+1}$  gehören und  $p, q$  werden getrennt.

3. Entweder stimmen  $\equiv_A^i$  und  $\equiv_A^{i+1}$  überein – und unser Algorithmus terminiert – oder mindestens eine Äquivalenzklasse von  $\equiv_A^i$  wird in zwei oder mehr Klassen zerlegt.
  - ▶  $\equiv_A^{i+1}$  besitzt mindestens eine Äquivalenzklasse mehr als  $\equiv_A^i$ .
  - ▶ Unser Algorithmus führt höchstens  $|Q| - 1$  Iterationen aus .
4. Man kann zeigen: Die Anzahl von Operationen pro Iteration ist höchstens proportional zur Anzahl  $|\Sigma| \cdot |Q|$  der Zustandübergänge.

Unser Algorithmus ist **effizient** und benötigt eine Anzahl von Operationen, die höchstens proportional zu  $|\Sigma| \cdot |Q|^2$  ist.

# Was ist zu tun?

Die wichtigen Fragen:

1. Sind  $A$  und der Äquivalenzklassenautomat  $A'$  äquivalent, d.h. berechnet  $A'$  dieselbe Sprache wie  $A$ ?
2. Ist  $A'$  minimal?

Aber zuerst rechnen wir zwei Beispiele durch.

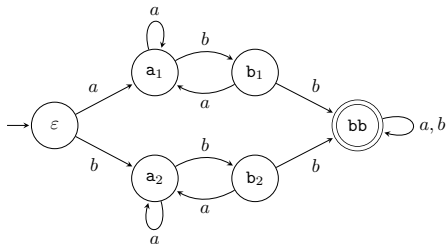


# Wie bestimmt man $M_{i+1}$ per Hand?

Es ist  $M_{i+1} = \{ \{p, q\} : p \equiv_A^i q, \text{ aber } \delta(p, a) \not\equiv_A^i \delta(q, a) \text{ f\"ur ein } a \in \Sigma \}$ .

- Betrachte jede gerade markierte Paarmenge  $\{r, s\} \in M_i$ .
- Eine noch nicht markierte Paarmenge  $\{p, q\}$  gehört genau dann zu  $M_{i+1}$ , wenn es einen Buchstaben  $a \in \Sigma$  gibt mit
  - ▶  $\delta(p, a) = r$  und  $\delta(q, a) = s$  oder
  - ▶  $\delta(p, a) = s$  und  $\delta(q, a) = r$ .

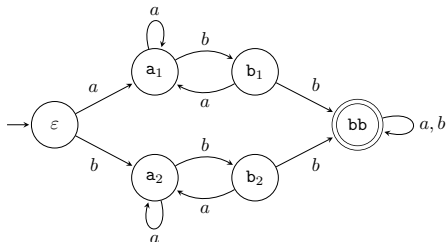
# Zustandsminimierung: Die Mengen $M_0, M_1, M_2$



$a_1$					
$a_2$					
$b_1$					
$b_2$					
$bb$					
	$\epsilon$	$a_1$	$a_2$	$b_1$	$b_2$

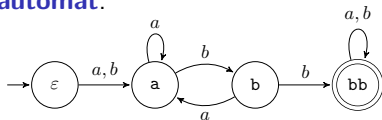
$a_1$				
$a_2$				
$b_1$				
$b_2$				

# Zustandsminimierung: Der Äquivalenzklassenautomat $A'$



$M_3 = \emptyset \Rightarrow \{bb\}$  ist die Klasse des einzigen akzeptierenden Zustands,  $\{\varepsilon\}$  ist die Klasse des Startzustands,  $\{a_1, a_2\}$  und  $\{b_1, b_2\}$  sind die restlichen Klassen.

Der **Äquivalenzklassenautomat**:



Genügt es, nur Zustände mit identischen Nachfolgezuständen zu verschmelzen?

**NEIN!**

# Zustandsminimierung: Die Tabelle

Wie haben wir die Tabelle gefüllt?

Angenommen, wir haben die Mengen  $M_0, \dots, M_i$  bestimmt und die „entsprechenden“ Positionen in der Tabelle markiert.

1. Dann haben wir nacheinander alle „**frisch markierten**“ Einträge  $\{r, s\} \in M_i$  inspiziert.
2. Wir haben die Position in Zeile  $p$  und Spalte  $q$  mit  $M_{i+1}$  markiert (bzw.  $p \neq^{i+1} q$  erhalten), wenn es einen Buchstaben  $a \in \Sigma$  gibt, so dass

$$(\delta(p, a) = r \text{ und } \delta(q, a) = s) \text{ oder } (\delta(p, a) = s \text{ und } \delta(q, a) = r),$$

d.h., so dass

$$\delta(p, a) \neq^i \delta(q, a)$$

(bzw.  $\{\delta(p, a), \delta(q, a)\} \in M_i$ ) gilt.

# Wie stellt man fest, ob ein Wort das Suffix $ab$ besitzt?

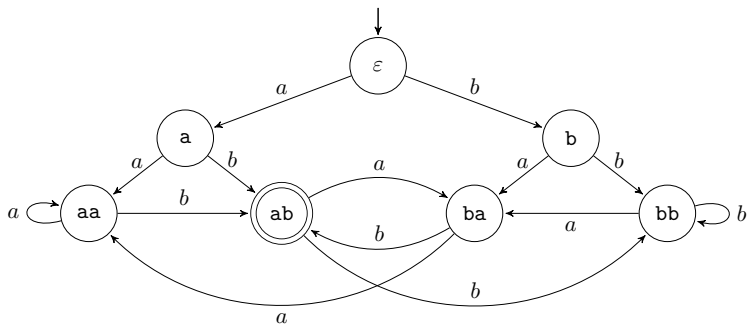
- Wir arbeiten mit dem Alphabet  $\Sigma = \{a, b\}$ .
- Speichere die beiden zuletzt gelesenen Buchstaben im aktuellen Zustand.
- Wir benutzen deshalb die Zustände

$$Q = \{\varepsilon, a, b, aa, ab, ba, bb\}$$

- ▶ mit Startzustand  $q_0 = \varepsilon$  („wir haben noch nichts gelesen“) und
- ▶ dem akzeptierenden Zustand  $ab$ : Die beiden letzten Buchstaben sind  $ab$ .

Welche Zustandsübergänge?

# Zustandsminimierung: Die Menge $M_0$



- (a) Der Automat  $A$  merkt sich tatsächlich die beiden letzten Buchstaben.
- (b)  $ab$  ist der einzige akzeptierende Zustand.

# Zustandsminimierung: Die Menge $M_0$

$ab$  ist der einzige akzeptierende Zustand  $\implies$  Die Menge  $M_0$  hat die Form

$$M_0 = \{\{\epsilon, ab\}, \{a, ab\}, \{b, ab\}, \{aa, ab\}, \{ba, ab\}, \{bb, ab\}\}$$

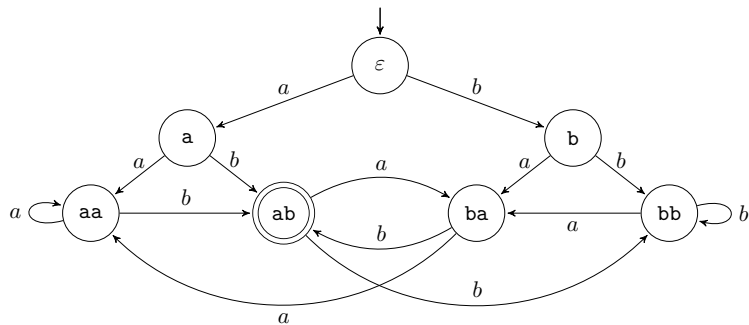
a						
b						
aa						
ab	$M_0$	$M_0$	$M_0$	$M_0$		
ba					$M_0$	
bb					$M_0$	
	$\epsilon$	a	b	aa	ab	ba

Was haben wir gelernt?

- (a)  $\{ab\}$  ist eine Klasse der Verschmelzungsrelation,
- (b) die restlichen Klassen sind disjunkte Teilmengen von  $\{\epsilon, a, b, aa, ba, bb\}$ .

Wie sieht  $M_1$  aus?

# Der Automat





# Zustandsminimierung: Die Menge $M_1$

Es ist  $M_0 = \{\{\epsilon, ab\}, \{a, ab\}, \{b, ab\}, \{aa, ab\}, \{ba, ab\}, \{bb, ab\}\}$ .

<i>a</i>						
<i>b</i>						
<i>aa</i>						
<i>ab</i>	$M_0$	$M_0$	$M_0$	$M_0$		
<i>ba</i>					$M_0$	
<i>bb</i>					$M_0$	
	$\epsilon$	<i>a</i>	<i>b</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>
<i>a</i>	$M_1$					
<i>b</i>		$M_1$				
<i>aa</i>						
<i>ab</i>	$M_0$	$M_0$	$M_0$	$M_0$		
<i>ba</i>					$M_0$	
<i>bb</i>		$M_1$			$M_0$	
	$\epsilon$	<i>a</i>	<i>b</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>
<i>a</i>	$M_1$					
<i>b</i>		$M_1$				

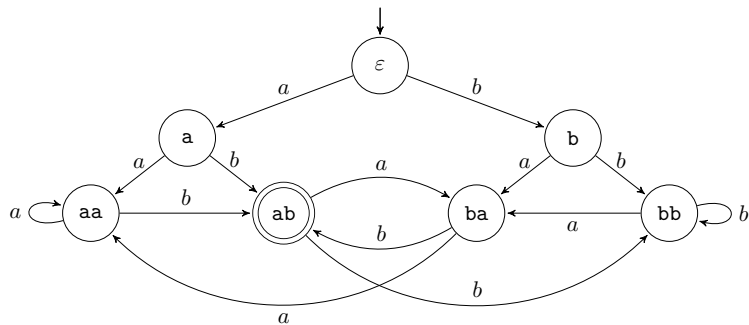
# Was haben wir gelernt?

<i>a</i>	$M_1$					
<i>b</i>		$M_1$				
<i>aa</i>	$M_1$		$M_1$			
<i>ab</i>	$M_0$	$M_0$	$M_0$	$M_0$		
<i>ba</i>	$M_1$		$M_1$		$M_0$	
<i>bb</i>		$M_1$		$M_1$	$M_0$	$M_1$
	$\epsilon$	<i>a</i>	<i>b</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>

1. Wir haben jeden der Zustände *a*, *aa*, *ba* von jedem der Zustände  $\epsilon$ , *b*, *bb* getrennt.
2. Neben der Äquivalenzklasse von *ab* sind die weiteren Äquivalenzklassen der Verschmelzungsrelation disjunkte Teilmengen
  - ▶ entweder von  $\{a, aa, ba\}$
  - ▶ oder von  $\{\epsilon, b, bb\}$ .

Können wir einen DFA aus den Klassen  $\{a, aa, ba\}$ ,  $\{\epsilon, b, bb\}$  und  $\{ab\}$  bauen?

# Der Automat



# Zustandsminimierung: Die Menge $M_2$

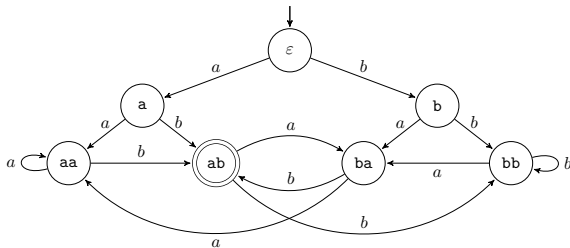
$a$	$M_1$					
$b$		$M_1$				
$aa$	$M_1$		$M_1$			
$ab$	$M_0$	$M_0$	$M_0$	$M_0$		
$ba$	$M_1$		$M_1$		$M_0$	
$bb$		$M_1$		$M_1$	$M_0$	$M_1$
	$\varepsilon$	$a$	$b$	$aa$	$ab$	$ba$

1. Die Zustände  $a$ ,  $aa$ ,  $ba$  können nicht voneinander getrennt werden, da sie alle unter  $a$  auf den Zustand  $aa$  und unter  $b$  auf den Zustand  $ab$  führen.
2.  $b$  und  $bb$  lassen sich ebenfalls nicht mehr trennen. Beide führen unter  $a$  auf  $ba$  und unter  $b$  auf  $bb$ .
3. Aber auch  $\varepsilon$  führt unter  $a$  auf die Klasse  $\{a, aa, ba\}$  und unter  $b$  auf  $b$ .

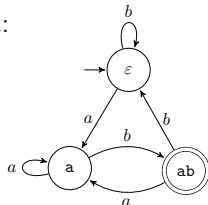
Es ist  $M_2 = \emptyset$ .

# Zustandsminimierung: Der Äquivalenzklassenautomat $A'$

- (a)  $\{ab\}$  ist die Klasse des einzigen akzeptierenden Zustands,
- (b)  $\{a, aa, ba\}$  und  $\{\varepsilon, b, bb\}$  sind die beiden verbleibenden Klassen.



Der **Äquivalenzklassenautomat**:



# Was ist Sache?

? Warum ist der ursprüngliche Automat  $A = (\Sigma, Q, \delta, q_0, F)$  mit dem **Äquivalenzklassenautomat**  $A' = (\Sigma, Q', \delta', q'_0, F')$  äquivalent?

- ▶ **Tafel:** Wenn  $p \equiv_A q$  und  $a \in \Sigma$ , dann gilt  $\delta(p, a) \equiv_A \delta(q, a)$ . ✓
- ▶ **Tafel:** Für alle Worte  $w \in \Sigma^*$  gilt:

$$\hat{\delta}(q, w) = r \implies \hat{\delta}'([q]_A, w) = [r]_A.$$

D.h.: Landet  $A$  im Zustand  $r$ , dann landet  $A'$  im Zustand  $[r]_A$ . ✓

- ▶ Jede Klasse der Verschmelzungsrelation hat entweder nur akzeptierende Zustände oder gar keine akzeptierenden Zustände. ✓

✓ Wir können  $A'$  effizient berechnen.

Aber es geht noch wesentlich schneller (siehe Theoretische Informatik 2):

Der Algorithmus von Hopcroft hat fast lineare Laufzeit und benötigt nur Zeit proportional zu  $|Q| \log |Q|$ .

? Aber hat  $A'$  unter allen mit  $A$  äquivalenten DFAs die kleinste Zustandszahl?

Die Nerode-Relation für die Sprache  $L = L(A)$  hat die Antwort.

# Die Nerode-Relation

Sei  $A = (\Sigma, Q, \delta, q_0, F)$  ein DFA.

Wenn  $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$ , dann  $\underbrace{\text{f.a. } w \in \Sigma^* : (uw \in L(A) \iff vw \in L(A))}_{\text{Eine Eigenschaft der Sprache } L(A)}$ .

Eine Eigenschaft der Sprache  $L(A)$

$L$  sei eine Sprache über der endlichen Menge  $\Sigma$ , d.h. es gilt  $L \subseteq \Sigma^*$ .

- (a) Die **Nerode-Relation**  $\equiv_L$  für  $L$  ist eine 2-stellige Relation **über**  $\Sigma^*$ .  
Für alle Worte  $u, v \in \Sigma^*$  definiere

$$u \equiv_L v : \iff \text{f.a. } w \in \Sigma^* \text{ gilt: } (uw \in L \iff vw \in L).$$

- (b) Wir sagen, dass das Wort  $w \in \Sigma^*$  die Worte  $u, v \in \Sigma^*$  **trennt**,  
bzw. dass  $w$  ein **Zeuge** für die Inäquivalenz von  $u$  und  $v$  ist, wenn

$$(uw \in L \wedge vw \notin L) \vee (uw \notin L \wedge vw \in L).$$

- (c) **Index(L)** ist die Anzahl der Äquivalenzklassen von  $\equiv_L$ .

# Die Nerode-Relation ist eine Äquivalenzrelation

Warum? Die Nerode-Relation  $\equiv_L$  ist

1. reflexiv, denn  $u \equiv_L u$  gilt f.a.  $u \in \Sigma^*$ ,
2. symmetrisch, denn aus  $u \equiv_L v$  folgt  $v \equiv_L u$  f.a.  $u, v \in \Sigma^*$ ,
3. transitiv, denn aus  $u \equiv_L v$ ,  $v \equiv_L w$  folgt  $u \equiv_L w$  f.a.  $u, v, w \in \Sigma^*$ .

**Beweis:** Wie im Fall der Verschmelzungsrelation  $\equiv_A$ .



Wie sehen die Äquivalenzklassen der Nerode-Relation in Beispielen aus?  
Insbesondere, wie groß ist  $\text{Index}(L)$ ?

1.  $L = \{w \in \{0, 1\}^* : w \text{ hat gerade viele Einsen}\}$ ,
2.  $L = \{a, b\}^* \{ab\}$ ,
3.  $L$  die Menge aller Binärdarstellungen für durch 6 teilbare Zahlen,
4.  $L_u = \{w \in \Sigma^* : u \text{ ist ein Teilwort von } w\}$  für ein Wort  $u \in \Sigma^*$ .

# Nerode-Klassen von $P = \{w \in \{0,1\}^* : w \text{ hat eine gerade Anzahl von Einsen}\}$

- (\*) Welche Wörter sind äquivalent mit  $\varepsilon$ ?
- (\*) Sei  $u$  ein beliebiges Wort mit gerade vielen Einsen und  $v$  ein beliebiges Wort mit ungerade vielen Einsen. Dann gilt

$$\begin{aligned}\varepsilon \equiv_P u &\iff \left( \text{f.a. } w \in \Sigma^* : \varepsilon w \in P \iff uw \in P \right) \\ &\iff \left( \text{f.a. } w \in \Sigma^* : w \in P \iff uw \in P \right) \iff u \in P.\end{aligned}$$

- (\*) Und die restlichen Äquivalenzklassen? Es gilt  $\varepsilon \not\equiv_P 1$  mit dem Zeugen  $\varepsilon$ .

- (a) Die Nerode-Relation für  $P$  hat genau zwei Äquivalenzklassen:

$$\begin{aligned}[\varepsilon]_P &= \{u \in \{0,1\}^* : u \text{ hat gerade viele Einsen}\} \text{ und} \\ [1]_P &= \{u \in \{0,1\}^* : u \text{ hat ungerade viele Einsen}\}.\end{aligned}$$

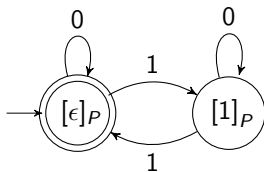
- (b) Es ist  $\text{Index}(P) = 2$  mit den Vertretern  $\varepsilon$  und  $1$ .

# Der Nerode-Automat $N_P$ für $P$

Für jedes Wort  $w \in \{0, 1\}^*$  soll  $N_P$  nach Lesen von  $w$  den Zustand  $[w]_P$  erreichen.

1.  $q_0 = [\epsilon]_P$  ist der Anfangszustand und  $q_0$  ist akzeptierend, denn  $\epsilon \in P$ .
2. Die Nachfolgezustände von  $q_0$ :
  - ▶ Der Nachfolgezustand unter 0 ist  $[\epsilon 0]_P = [0]_P = [\epsilon]_P$ .
  - ▶ Der Nachfolgezustand unter 1 ist dann die Klasse  $[1]_P$ .
3. Der Zustand  $[1]_P$  ist nicht akzeptierend, denn  $1 \notin P$ .
  - ▶ Lesen wir im Zustand  $[1]_P$  eine 0, dann erreichen wir die Äquivalenzklasse von 10, aber  $[10]_P = [1]_P$ .
  - ▶ Lesen wir eine 1, dann erreichen wir die Äquivalenzklasse von 11, aber  $[11]_P = [\epsilon]_P$ .

Hier ist der **Nerode-Automat**  $N_L$ :



# Worauf ist beim Bau des Nerode-Automaten zu achten?

Sei  $L$  eine beliebige Sprache über dem Alphabet  $\Sigma$  und  $a \in \Sigma$  ein beliebiger Buchstabe. Dann gilt **für beliebige** Worte  $u, w \in \Sigma^*$

$$u \equiv_L w \implies ua \equiv_L wa.$$

**Beweis:** Angenommen  $u \equiv_L w$  gilt, aber ebenso  $ua \not\equiv_L wa$ .

$\implies$  Dann gibt es einen Zeugen  $v \in \Sigma^*$  mit z.B.  $uav \in L$ , aber  $wav \notin L$ .

$\implies$  Aber dann folgt  $u \not\equiv_L w$  mit dem Zeugen  $av$ , im Widerspruch zur Annahme  $\zeta$

Wenn wir den Nerode-Automaten  $N_L$  aus den Nerode-Klassen bauen:

1. Beginne die Konstruktion mit der Klasse  $[\varepsilon]_L$  als Startzustand.
2. Unser Ziel:  $N_L$  soll nach Lesen von  $w$  den Zustand  $[w]_L$  erreichen.  
 $\implies$  Für jede Klasse  $[u]_L$  und jeden Buchstaben  $a \in \Sigma$  wähle den Übergang

$$\delta_{N_L}([u]_L, a) := [ua]_L.$$

# Der Nerode-Automat $N_S$ der Suffixsprache $S = \{a, b\}^* \cdot ab$

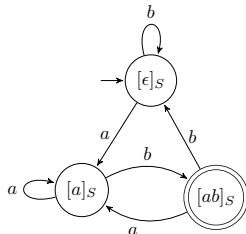
- Der Startzustand ist die Klasse des leeren Worts, also  $q_0 = [\epsilon]_S$  und es gilt

$$[\epsilon]_S = \{\epsilon, b\} \cup \{ubb : u \in \{a, b\}^*\}.$$

Die Nachfolgezustände von  $q_0$ :

- ▶  $\delta_{N_S}(q_0, a) := [a]_S$ . Es ist  $\epsilon \not\equiv_S a$  mit dem Zeugen  $v = b$ .
- ▶  $\delta_{N_S}(q_0, b) := [b]_S = q_0$ , denn  $\epsilon \equiv_S b$  gilt.
- Es ist  $[a]_S = \{ua : u \in \{a, b\}^*\}$ . Die Nachfolgezustände von  $[a]_S$ :
  - ▶ Setze  $\delta_{N_S}([a]_S, a) := [aa]_S = [a]_S$ , denn  $aa \equiv_S a$ .
  - ▶ Setze  $\delta_{N_S}([a]_S, b) := [ab]_S$ . Es gilt  $[ab]_S = S$  und  $[ab]_S$  ist akzeptierend.
- Die Nachfolgezustände von  $[ab]_S$ :
  - ▶ Setze  $\delta_{N_S}([ab]_S, a) := [aba]_S = [a]_S$  und  $\delta_{N_S}([ab]_S, b) := [abb]_S = [\epsilon]_S$ .

Der **Nerode-Automat**  $N_S$ :



Sei  $L$  eine Sprache über dem Alphabet  $\Sigma$ . Dann hat der **Nerode-Automat**

$$N_L = (Q, \Sigma, \delta, q_0, F)$$

die folgenden Komponenten:

- Die Zustandsmenge  $Q$  besteht aus allen Äquivalenzklassen der Nerode-Relation  $\equiv_L$ .
- Es ist

$$\delta([w]_L, a) := [wa]_L$$

für jedes Wort  $w \in \Sigma^*$  und jeden Buchstaben  $a \in \Sigma$ .

- $q_0 := [\epsilon]_L$  ist der Anfangszustand.
- $F := \{ [w]_L : w \in L \}$  ist die Menge der akzeptierenden Zustände.

Sei  $L$  eine Sprache über dem Alphabet  $\Sigma$ .

- (a) Für jedes Wort  $w \in \Sigma^*$  erreicht der Nerode-Automat  $N_L$  den Zustand  $[w]_L$ , d.h.

$$\widehat{\delta}_{N_L}([\epsilon]_L, w) = [w]_L.$$

- (b)  $N_L$  akzeptiert die Sprache  $L$  mit  $\text{Index}(L)$  vielen Zuständen.

Teil (a) zeigt man durch vollständige Induktion über die Länge von  $w$ .

INDUKTIONSANFANG: Für  $w = \epsilon$  ist  $\widehat{\delta}_{N_L}(q_0, w) = q_0 := [\epsilon]_L$ . ✓

INDUKTIONSSCHRITT: Für  $w = ua$  ist

$$\widehat{\delta}_{N_L}(q_0, w) = \widehat{\delta}_{N_L}(q_0, ua) = (\delta_{N_L}(\widehat{\delta}_{N_L}(q_0, u), a) \stackrel{\text{Ind. Vor.}}{=} \delta_{N_L}([u]_L, a) \stackrel{\text{Def.}}{=} [ua]_L. \checkmark$$

Für Teil (b): Gibt es ein Wort  $u \in L$  und ein Wort  $w \notin L$  mit  $u \equiv_L w$ ? Kann nicht passieren, denn dann trennt der Zeuge  $v = \epsilon$  die Worte  $u$  und  $w$ .

**Zusammengefasst:**  $N_L$  erreicht den Zustand  $[u]_L$  für das Wort  $u$ . Also folgt  $[u]_L \in F \iff u \in L$ .

# Der Satz von Myhill-Nerode I

Sei  $A = (\Sigma, Q, \delta, q_0, F)$  ein DFA und sei  $L = L(A)$ .

- (a)  $\text{Index}(L)$  = minimale Zustandszahl eines DFA für die Sprache  $L$ .
- (b) Der Äquivalenzklassenautomat  $A'$  wie auch der Nerode-Automat ist minimal.

**WOW**

(a) Für Teil (a) gelte  $k = \text{Index}(L)$ . Der DFA  $A$  akzeptiere  $L$ .

- ▶  $u_1, \dots, u_k$  seien Vertreter der  $k$  Nerode-Klassen.
- ▶ Angenommen, zwei dieser Vertreter, z.B.  $u_i$  und  $u_j$  (mit  $i \neq j$ ), führen auf denselben Zustand in  $Q$ 
  - $\implies \widehat{\delta}(q_0, u_i) = \widehat{\delta}(q_0, u_j)$
  - $\implies \widehat{\delta}(q_0, u_i w) = \widehat{\delta}(q_0, u_j w)$  für alle Worte  $w \in \Sigma^*$ .
  - $\implies u_i w \in L \iff u_j w \in L$  für alle Worte  $w \in \Sigma^*$  und  $u_i \equiv_L u_j$  folgt ⚡.
- ▶ Es ist  $|Q| \geq \text{Index}(L)$ .

(b) Der Nerode-Automat  $N_L$  akzeptiert  $L$  mit genau  $\text{Index}(L)$  vielen Zuständen  
 $\implies$  Der Nerode-Automat ist minimal.

Aber warum ist der Äquivalenzklassenautomat minimal?



# Der Äquivalenzklassenautomat ist minimal

Der DFA  $A = (Q, \Sigma, \delta, q_0, F)$  akzeptiere die Sprache  $L$ , es gelte also  $L(A) = L$ .  
 $A' = (Q', \Sigma, \delta', q'_0, F')$  sei sein Äquivalenzklassenautomat.

Angenommen, Worte  $u, v \in \Sigma^*$  führen in  $A'$  zu Zuständen  $[p]_A \neq [q]_A$ .

1. Für  $A$  sei  $p = \widehat{\delta}(q_0, u)$  und  $q = \widehat{\delta}(q_0, v)$ .
2. Es folgt  $p \not\equiv_A q$  und es gibt einen Zeugen  $w \in \Sigma^*$  für die Nicht-Äquivalenz.
  - ▶ Also:  $(\widehat{\delta}(p, w) \in F \wedge \widehat{\delta}(q, w) \notin F) \vee (\widehat{\delta}(p, w) \notin F \wedge \widehat{\delta}(q, w) \in F)$ , bzw.
  - ▶  $(\widehat{\delta}(q_0, uw) \in F \wedge \widehat{\delta}(q_0, vw) \notin F) \vee (\widehat{\delta}(q_0, uw) \notin F \wedge \widehat{\delta}(q_0, vw) \in F)$ , bzw.
  - ▶  $(uw \in L \wedge vw \notin L) \vee (uw \notin L \wedge vw \in L)$  bzw.  $u \not\equiv_L v$ .
3. Wenn  $u$  und  $v$  in  $A'$  zu verschiedenen Zuständen führen, dann folgt  $u \not\equiv_L v$ .
4. Die Zustandszahl von  $A'$  ist höchstens so groß wie  $\text{Index}(L)$

Die **Zustandszahl von  $A'$**  stimmt überein mit  **$\text{Index}(L(A))$** :

**Der Äquivalenzklassenautomat  $A'$  ist ein minimaler DFA für  $L(A)$ .**

# Reguläre Sprachen

Eine Teilmenge  $L \subseteq \Sigma^*$  heißt eine **reguläre Sprache**, wenn es einen DFA  $A$  gibt mit

$$L = L(A).$$

- (a) Ist  $L = \{ a^n b^n : n \in \mathbb{N} \}$  eine reguläre Sprache?
- (b) Sei  $\Sigma$  ein beliebiges Alphabet und sei  $w \in \Sigma^*$  ein Wort über  $\Sigma$ . Ist

$$L = \{ u \in \Sigma^* : w \text{ ist ein Teilwort von } u \}$$

eine reguläre Sprache?

## Der Satz von Myhill-Nerode II

## Satz von Myhill-Nerode II

Eine Teilmenge  $L \subseteq \Sigma^*$  ist **regulär**  $\iff$  **Index(L) ist endlich.**

$\implies$   $L \subseteq \Sigma^*$  sei regulär. Dann gibt es einen DFA  $A$  mit  $L = L(A)$ .

- ▶  $A$  hat endlich viele Zustände, aber mindestens  $\text{Index}(L)$  Zustände.
- ▶ Also ist  $\text{Index}(L)$  endlich.

$\impliedby$   $\text{Index}(L)$  sei endlich.

- ▶ Der Nerode-Automat  $N_L$  ist ein DFA.
- ▶ Für  $N_L$  gilt  $L(N_L) = L \implies L$  ist regulär.

$L = \{a^n b^n : n \in \mathbb{N}\}$  ist nicht regulär, weil ...

**Zeige:**  $\text{Index}(L) = \infty$ , also bestimme unendlich viele Worte  $u_k \in \{a, b\}^*$ , so dass

$$u_k \not\equiv_L u_\ell$$

für alle  $k \neq \ell$  gilt.

Setze  $u_k := a^k$ . Für  $k \neq \ell$  gilt  $u_k \not\equiv_L u_\ell$ , denn der Zeuge  $w = b^k$  trennt  $u_k$  und  $u_\ell$ :

$$a^k b^k \in L, \text{ aber } a^\ell b^k \notin L.$$

$\text{Index}(L) = \infty$  und  $L$  ist nicht regulär.

DFAs können nicht (unbeschränkt) zählen.

$L = \{ww : w \in \{a, b\}^*\}$  ist nicht regulär, weil ...

**Zeige:**  $\text{Index}(L) = \infty$ , also bestimme unendlich viele Worte  $u_k \in \{a, b\}^*$ , so dass

$$u_k \not\equiv_L u_\ell$$

für alle  $k \neq \ell$  gilt.

Setze  $u_k := a^k$ . Für  $k \neq \ell$  gilt  $u_k \not\equiv_L u_\ell$ , denn der Zeuge  $w = ba^k b$  trennt  $u_k, u_\ell$ :

$$a^k ba^k b \in L, \text{ aber } a^\ell ba^k b \notin L.$$

$\text{Index}(L) = \infty$  und  $L$  ist nicht regulär.

DFAs können sich nur beschränkt viele Dinge merken.

Keine der folgenden Sprachen ist regulär.

- $L_1 = \{ a^n b^m : n, m \in \mathbb{N}, n \leq m \}$ .
- $L_2 = \{ a^n b^m c^{n+m} : n, m \in \mathbb{N} \}$ .
- $L_3 = \{ a^{n^2} : n \in \mathbb{N} \}$ .
- $L_4 = \{ w \in \{a, b\}^* : w \text{ ist ein Palindrom} \}$ :



Zeige jeweils, dass der Index unendlich ist.

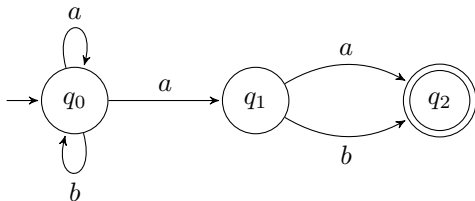


# NFAs

# NFAs: DFAs, die raten dürfen

Ein „NFA“ akzeptiert ein Eingabewort  $w \in \{a, b\}^*$  **genau dann**, wenn es im Zustandsdiagramm **mindestens einen Weg gibt**,

- der im Startzustand  beginnt,
- dessen Kanten mit  $w$  beschriftet sind,
- und der in einem akzeptierenden Zustand  endet.



Der „NFA“ akzeptiert

$$L = \{w \in \{a, b\}^* : \text{der vorletzte Buchstabe von } w \text{ ist ein } a \}.$$

Ein nichtdeterministischer endlicher Automat (kurz: **NFA**)

$$A = (\Sigma, Q, \delta, q_0, F)$$

besteht aus:

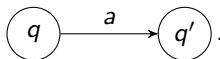
- einer endlichen Menge  $\Sigma$ , dem Eingabealphabet,
- einer endlichen Menge  $Q$ , der Zustandsmenge,
- dem Startzustand  $q_0 \in Q$ ,
- einer Menge  $F \subseteq Q$  von Endzuständen bzw. akzeptierenden Zuständen und
- einer Funktion  $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ , der Übergangsfunktion,
  - die jedem Zustand  $q \in Q$  und jedem Symbol  $a \in \Sigma$  eine **Menge**  $\delta(q, a)$  von **möglichen** Nachfolgezuständen zuordnet.
  - Möglicherweise ist  $\delta(q, a) = \emptyset$ : Dann „stürzt“ der Automat ab, wenn er im Zustand  $q$  ist und das Symbol  $a$  liest.

# Das Zustandsdiagramm von NFAs

Es sei

- $q \in Q$  ein Zustand,
- $a \in \Sigma$  ein Eingabesymbol und
- $q' \in \delta(q, a)$  ein „möglicher Nachfolgezustand“.

Dann gibt es im Zustandsdiagramm einen mit dem Symbol  $a$  beschrifteten Pfeil von Knoten  $q$  zu Knoten  $q'$ , d.h.

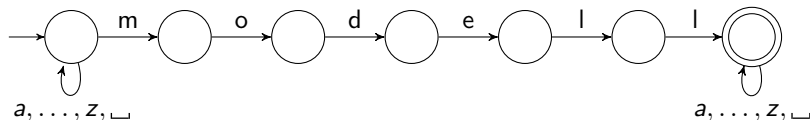


*Gegeben:* Ein Stichwort, z.B. „modell“

*Eingabe:* Ein Text, der aus den Buchstaben „a“ bis „z“ sowie dem Leerzeichen „ $\sqcup$ “ besteht

*Frage:* Kommt das Stichwort „modell“ irgendwo im Eingabetext vor?

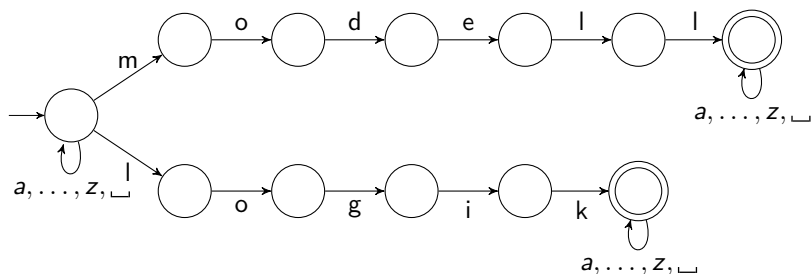
Das Zustandsdiagramm eines NFAs, der dies bewerkstelligt:



Auf ähnliche Art können auch Varianten dieser Stichwortsuche behandelt werden, zum Beispiel die Frage

*Kommt mindestens eins der Stichworte „modell“ oder „logik“ im Eingabetext vor?*

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



# Die von einem NFA akzeptierte Sprache

# Wann akzeptiert ein NFA?

Sei  $A = (\Sigma, Q, \delta, q_0, F)$  ein NFA.

- (a) Sei  $n \in \mathbb{N}$  und sei  $w = a_1 \cdots a_n$  ein Eingabewort der Länge  $n$ . Das Wort  $w$  wird genau dann vom NFA  $A$  **akzeptiert**, wenn es im Zustandsdiagramm von  $A$  einen im Startzustand



beginnenden Weg der Länge  $n$  gibt, dessen Kanten mit den Symbolen  $a_1 \dots a_n$  beschriftet sind und der in einem akzeptierenden Zustand endet.

- (b) Die von  $A$  akzeptierte Sprache  $L(A)$  ist

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

Das ist keine „wirklich“ präzise Definition, denn das Zustandsdiagramm soll doch nur unsere Intuition unterstützen.



# Die erweiterte Übergangsfunktion

$\widehat{\delta}(q, w)$  := die MENGE aller möglichen Zustände nach Lesen von  $w$  im „Startzustand“  $q$

Sei  $A := (\Sigma, Q, \delta, q_0, F)$  ein NFA. Die Funktion

$$\widehat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$$

ist rekursiv wie folgt definiert:

- **Rekursionsanfang:** F.a.  $q \in Q$  ist  $\widehat{\delta}(q, \varepsilon) := \{q\}$ .
- **Rekursionsschritt:** F.a.  $q \in Q$ ,  $w \in \Sigma^*$  und  $a \in \Sigma$  ist

$$\widehat{\delta}(q, wa) := \bigcup_{q' \in \widehat{\delta}(q, w)} \delta(q', a).$$

Ein möglicher Zustand  $q''$  wird nach Lesen von  $wa$  genau dann erreicht, wenn nach Lesen von  $w$  (im Zustand  $q$ ) ein Zustand  $q'$  erreicht wird und  $q'' \in \delta(q', a)$  gilt:

$$q \xrightarrow{w} q' \xrightarrow{a} q''.$$

# Wann genau akzeptiert denn nun ein NFA?

Der NFA  $A = (\Sigma, Q, \delta, q_0, F)$  akzeptiert ein Wort  $w$  genau dann, wenn:

$$\widehat{\delta}(q_0, w) \cap F \neq \emptyset.$$

Somit ist

$$L(A) = \{ w \in \Sigma^* : \widehat{\delta}(q_0, w) \cap F \neq \emptyset \}.$$

# Äquivalenz von NFAs und DFAs

# Sind NFAs mächtiger als DFAs?

Können NFAs Sprachen akzeptieren, die DFAs nicht akzeptieren können?

# NEIN!

Für jeden NFA  $A = (\Sigma, Q, \delta, q_0, F)$  gibt es einen DFA  $A' = (\Sigma, Q', \delta', q'_0, F')$  mit

$$L(A') = L(A).$$

D.h.: NFAs und DFAs akzeptieren genau dieselben Sprachen.

D.h. wir können Stichwortsuche auch mit DFAs durchführen?

- Natürlich und sogar mit genau so vielen Zuständen.

Aber im Allgemeinen sind DFAs doch bestimmt sehr viel größer?!?

Sei  $A = (\Sigma, Q, \delta, q_0, F)$  der gegebene NFA.

**Idee:** Wir konstruieren einen DFA  $A' = (\Sigma, Q', \delta', q'_0, F')$ , der in seinem aktuellen Zustand  $q' \in Q'$

die **Menge** aller Zustände abspeichert, in denen der Automat  $A$  in der aktuellen Situation sein **könnte**.

Wir definieren die Komponenten von  $A'$  daher wie folgt:

- Eingabealphabet  $\Sigma$ ,
- Zustandsmenge  $Q' := \mathcal{P}(Q)$ ,
- Startzustand  $q'_0 := \{q_0\}$ ,
- Endzustandsmenge  $F' := \{X \in Q' : X \cap F \neq \emptyset\}$ ,
- Übergangsfunktion  $\delta' : Q' \times \Sigma \rightarrow Q'$ ,  
wobei für alle  $X \in Q'$  und alle  $a \in \Sigma$  gilt:

$$\delta'(X, a) := \bigcup_{q \in X} \delta(q, a).$$

Und wie zeigt man, dass  $A$  und  $A'$  dieselbe Sprache akzeptieren?

Zeige für jede Eingabe  $w \in \Sigma^*$ , dass sich der DFA  $A'$  stets in der Menge aller Zustände befindet, in denen der NFA sein könnte. D.h. zeige dass gilt:

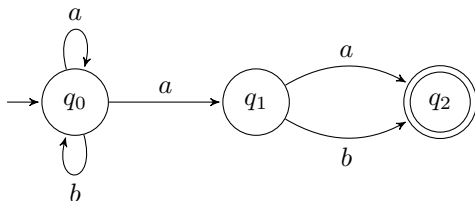
$$\widehat{\delta}'(\{q_0\}, w) = \widehat{\delta}(q_0, w).$$

Und wie, bitte schön, sollen wir das zeigen?

- Wir haben die erweiterten Übergangsfunktionen  $\widehat{\delta}$  und  $\widehat{\delta}'$  rekursiv definiert.
- Dann werden wir wohl eine vollständige Induktion nach  $n = |w|$  ausführen!

Beweis im Skript.

Wie führt man die Potenzmengenkonstruktion für den NFA

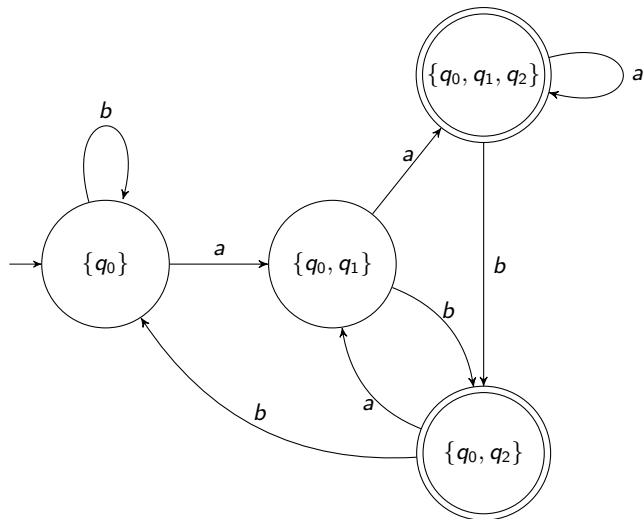


aus?

**Wichtig, Wichtig, Wichtig, Wichtig, Wichtig, Wichtig, Wichtig, Wichtig**

1. Bestimme alle möglichen Nachfolgezustände des Startzustands  $q'_0 := \{q_0\}$  und wiederhole das Vorgehen für die Nachfolger ...
2. Also: Definiere die Übergangsfunktion von  $A'$  nur für solche Zustände  $X \in \mathcal{P}(\{q_0, q_1, q_2\})$ , die vom Startzustand  $q'_0$  aus erreicht werden können.





# Reguläre Ausdrücke

Sei  $\Sigma$  ein Alphabet.

- 1 Die Menge aller Worte über  $\Sigma$  haben wir beschrieben mit

$$\Sigma^*.$$

- 2 Sei  $u$  ein Wort über  $\Sigma$ . Dann wird die Menge aller Worte über  $\Sigma$ , die  $u$  als Teilwort besitzen, beschrieben durch

$$\Sigma^* \cdot \{u\} \cdot \Sigma^*.$$

- 3 Die Menge aller Worte über  $\Sigma$ , deren vorletzter Buchstabe ein  $a$  ist, wird beschrieben durch:

$$\Sigma^* \cdot \{a\} \cdot \Sigma$$

# Reguläre Ausdrücke: Eine rekursive Definition der Syntax

Sei  $\Sigma$  ein endliches Alphabet.

Die Menge aller **regulären Ausdrücke** über  $\Sigma$  ist rekursiv wie folgt definiert:

## Basisregeln:

- $\emptyset$  ist ein regulärer Ausdruck über  $\Sigma$  („leere Menge“).
- $\varepsilon$  ist ein regulärer Ausdruck über  $\Sigma$  („leeres Wort“).
- Für jedes  $a \in \Sigma$  gilt:  $a$  ist ein regulärer Ausdruck über  $\Sigma$ .

## Rekursive Regeln:

- Ist  $R$  ein regulärer Ausdruck über  $\Sigma$ ,  
so ist auch  $R^*$  ein regulärer Ausdruck über  $\Sigma$  („Kleene-Stern“).
- Sind  $R$  und  $S$  reguläre Ausdrücke über  $\Sigma$ , so ist auch
  - ▶  $(R \cdot S)$  ein regulärer Ausdruck über  $\Sigma$  („Konkatenation“).
  - ▶  $(R \mid S)$  ein regulärer Ausdruck über  $\Sigma$  („Vereinigung“).

Wir haben gerade **formal definiert**,

„was ein regulärer Ausdruck  $R$  ist“

Aber was

„**bedeutet**“  $R$ ?

$R$  sollte für eine Sprache stehen!

Sei  $\Sigma$  ein endliches Alphabet. Jeder reguläre Ausdruck  $R$  über  $\Sigma$  **beschreibt** (oder definiert) eine Sprache  $L(R) \subseteq \Sigma^*$ , die induktiv wie folgt definiert ist:

## Basisregeln:

- $L(\emptyset) := \emptyset$ .
- $L(\varepsilon) := \{\varepsilon\}$ .
- Für jedes  $a \in \Sigma$  gilt:  $L(a) := \{a\}$ .

## Rekursive Regeln:

- Ist  $R$  ein regulärer Ausdruck über  $\Sigma$ , so ist

$$\begin{aligned} L(R^*) &:= \{\varepsilon\} \cup \{w_1 \cdots w_k : k \in \mathbb{N}_{>0}, w_1 \in L(R), \dots, w_k \in L(R)\} \\ &= (L(R))^* \end{aligned}$$

- Sind  $R$  und  $S$  reguläre Ausdrücke über  $\Sigma$ , so ist
  - ▶  $L((R \cdot S)) := \{wu : w \in L(R), u \in L(S)\} = L(R) \cdot L(S)$ .
  - ▶  $L((R \mid S)) := L(R) \cup L(S)$ .

# Reguläre Ausdrücke: Vereinfachte Schreibweise

Zur vereinfachten Schreibweise und besseren Lesbarkeit regulärer Ausdrücke:

- Den „Punkt“ bei der Konkatination

$$(R \cdot S)$$

darf man weglassen.

- Bei Ketten gleichartiger Operatoren verzichten wir auf Klammern:

$$(R_1 | R_2 | R_3 | R_4) \text{ statt } \left( \left( (R_1 | R_2) | R_3 \right) | R_4 \right) \text{ und}$$

$$(R_1 R_2 R_3 R_4) \text{ statt } \left( \left( (R_1 R_2) R_3 \right) R_4 \right).$$

- „Präzedenzregeln“:
  - \* bindet stärker als ·
  - bindet stärker als |
- Äußere Klammern, die einen regulären Ausdruck umschließen, dürfen weggelassen werden. Zusätzliche Klammern dürfen eingeführt werden.

- (a)  $a | bc^*$  ist eine verkürzte Schreibweise für  $(a | (b \cdot c^*))$ .

Die von diesem regulären Ausdruck beschriebene Sprache  $L = L(a | bc^*)$  ist

$$L = \{a\} \cup \{w \in \{a, b, c\}^* : \text{der erste Buchstabe von } w \text{ ist ein } b \text{ und} \\ \text{alle weiteren Buchstaben von } w \text{ sind } c\text{'s}\}.$$

- (b)  $L((a | b)^*) = \{a, b\}^*$ .

- (c) Die Menge aller Worte über  $\{a, b, c\}$ , in denen  $abb$  als Teilwort vorkommt, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a | b | c)^* abb (a | b | c)^*.$$

- (d) Die Menge aller Worte über  $\{a, b, c\}$ , deren letzter oder vorletzter Buchstabe ein  $a$  ist, wird durch den folgenden regulären Ausdruck beschrieben:

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



Wir wollen einen regulären Ausdruck angeben, der alle Telefonnummern der Form

*Vorwahl – Nummer*

beschreibt, wobei

- 1 *Vorwahl* und *Nummer* nicht-leere Ziffernfolgen sind,
- 2 *Vorwahl* mit genau einer Null beginnt und *Nummer* nicht mit einer Null beginnt.

Der Ausdruck

$$0 (1|2|\dots|9) (0|1|\dots|9)^* - (1|2|\dots|9) (0|1|\dots|9)^*$$

definiert die gewünschte Sprache.

Der Ausdruck:

$$R := (\varepsilon \mid 069 -) \ 798 \ (\varepsilon \mid -) \ (0 \mid (1|2|\dots|9) (0|1|\dots|9)^*)$$

definiert eine Sprache. Welche der folgenden Worte gehören zu  $R$ ?

? 069 - 798 - 0

? 7980

? 069 - 798 - 028551.

# Reguläre Sprachen und reguläre Ausdrücke

Jeder reguläre Ausdruck  $R$  definiert eine reguläre Sprache.

Beweis durch Induktion über den Aufbau von  $R$ : Siehe Skript/Übungen.

Die Klasse der regulären Sprache ist ein fundamentales Konzept mit verschiedensten Sichtweisen, denn

DFAs, NFAs oder reguläre Ausdrücke

definieren **dieselben** Sprachen!

Dieses Ergebnis wird in der Veranstaltung „**Theoretische Informatik 2**“ gezeigt. Insbesondere besitzt also jede reguläre Sprache einen regulären Ausdruck!

In der Veranstaltung „**Theoretische Informatik 2**“ wird unter Anderem gezeigt:

- (a) dass auch „**Zweiweg-Automaten**“ und „**reguläre Grammatiken**“ die Klasse der regulären Sprachen definieren,
- (b) und dass – unter bestimmten Umständen – auch **würfelnde Automaten** genau die Klasse der regulären Sprachen beschreiben,
- (c) dass viele Entscheidungsfragen wie
  - ▶ akzeptieren zwei DFAs dieselbe Sprache?
  - ▶ akzeptiert ein NFA mindestens ein Wort?effizient beantwortet werden können, andere hingegen, wie etwa
  - ▶ akzeptieren zwei NFAs dieselbe Sprache?
  - ▶ akzeptiert ein NFA alle Worte eines Alphabets?viel zu schwierig sind!

- (a) Ein äquivalenter DFA mit der kleinstmöglichen Zustandszahl kann effizient bestimmt werden.
- ▶ Für einen DFA  $A$  haben wir die **Verschmelzungsrelation**  $\equiv_A$  und den **Äquivalenzklassenautomaten**  $A'$  berechnet.
  - ▶ Die **Nerode-Relation**  $\equiv_L$  für eine Sprache  $L$  hat uns geholfen, den **Nerode-Automaten**  $N_L$  zu definieren.
  - ▶ Der **Satz von Myhill-Nerode I**: Es gelte  $L = L(A)$  für einen DFA  $A$ . Dann ist der Äquivalenzklassenautomat  $A'$  wie auch der Nerode-Automat  $N_L$  minimal.  $\implies$   $\text{Index}(L)$  ist die minimale Zustandszahl eines mit  $A$  äquivalenten DFA.
  - ▶ Der **Satz von Myhill-Nerode II**:  $L$  ist nicht regulär  $\iff \text{Index}(L) = \infty$ .
- (b) **NFAs** sind Automaten, die raten dürfen.
- ▶ Potenzmengenkonstruktion: Zu jedem NFA gibt es einen äquivalenten DFA.
  - ▶ Achtung: Die Zustandszahl des DFA kann exponentiell größer sein!
- (c) **Reguläre Ausdrücke** ist ein weiterer Formalismus, den jeder Profi kennt.
- ▶ Zu jedem DFA gibt es einen regulären Ausdruck und umgekehrt.