Endliche Automaten

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**

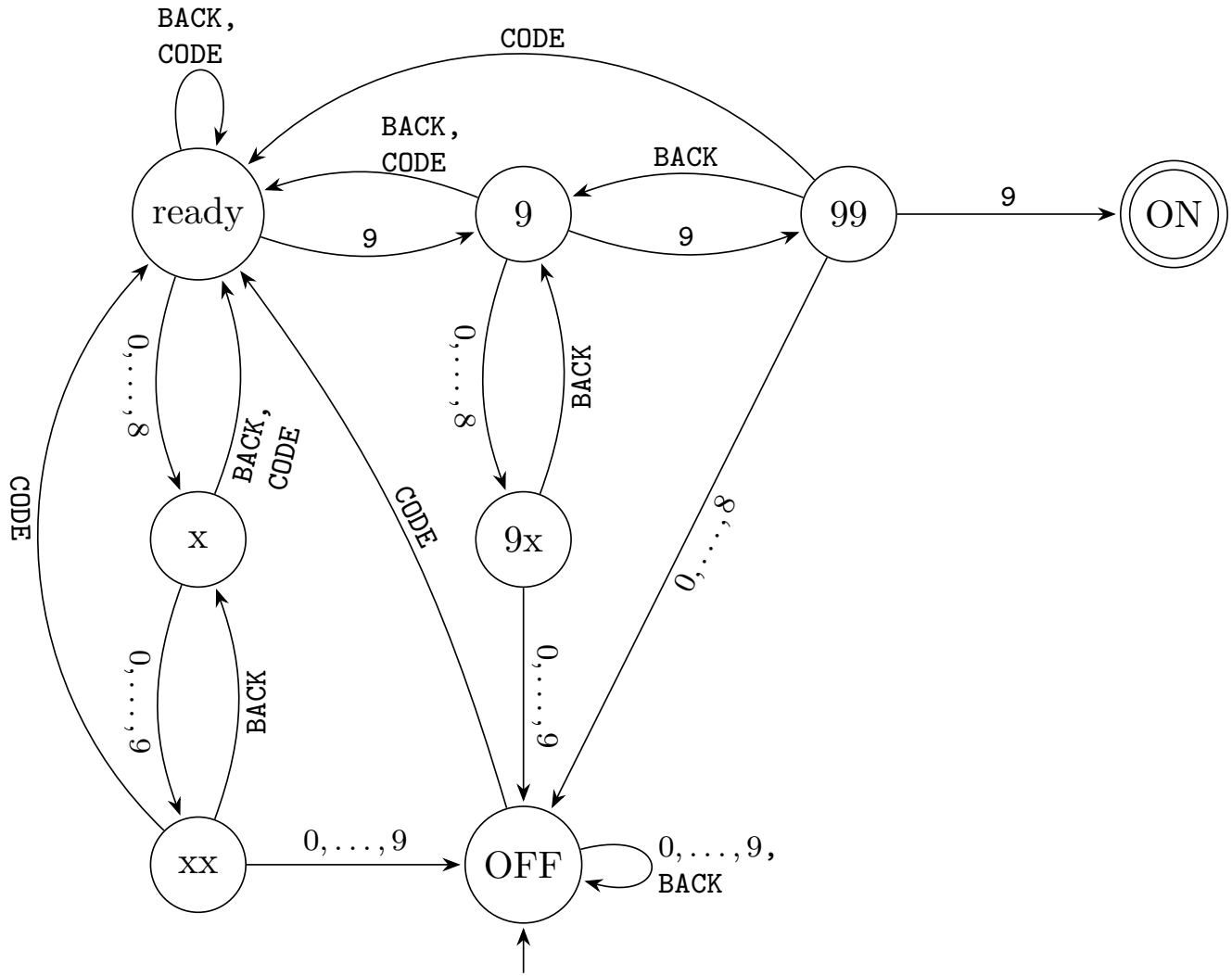
  also der Menge aller Folgen von Ereignissen,
  die von einem Startzustand in einen gewünschten Zustand führen,
- in der **Entwicklung digitaler Schaltungen**
- in der **Softwaretechnik** (z. B. in der Modellierung des Applikationsverhaltens)
- in der **Compilierung**: Lexikalische Analyse
- im **Algorithmenentwurf**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

Freischaltung eines Fernsehers (1/2)

- Um die Kindersicherung des Fernsehers über die Fernbedienung freizuschalten, muss ein dreistelliger Code korrekt eingegeben werden. Dabei sind die folgenden Tasten relevant:
  - Die Tasten $0, \ldots, 9$,
  - 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.

Freischaltung eines Fernsehers (2/2)



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

Alphabete, Worte und Sprachen

Worte und Alphabete

Sei Σ eine endliche Menge: Wir nennen Σ ein **Alphabet**.

- Wir fassen ein Tupel
$$w = (a_1, \ldots, a_k) \in \Sigma^k$$
als ein **Wort** mit den „Buchstaben“ $a_1, \ldots, a_k$ 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 ε (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 Σ, so ist die **Konkatenation** von v und w das Wort
$$vw := a_1 \cdots a_k b_1 \cdots b_\ell.$$

Sprachen

Sei Σ ein Alphabet. (Also ist Σ eine Menge.)

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

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

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

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

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

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

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

Natürliche Sprachen

Wir betrachten das Alphabet

$$\Sigma_{\text{deutsch}} := \{\text{A}, \text{B}, \ldots, \text{Z}, \text{Ä}, \text{Ö}, \text{Ü}, \text{a}, \text{b}, \ldots, \text{z}, \text{ä}, \text{ö}, \text{ü}, \text{ß}, \text{.}, \text{,}, \text{:}, \text{;}, \text{!}, \text{?}, \text{-}, \text{\_}\}.$$

Beispiele für Sprachen über Σ_{deutsch} sind:

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

$$L_2^+.$$

Programmiersprachen

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

$$\text{ASCII} \;:=\; \text{die Menge aller ASCII-Symbole.}$$

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.

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

Freischaltung eines Fernsehers

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

$$\Sigma := \{0, 1, \ldots, 9\} \cup \{\texttt{CODE}, \texttt{BACK}\},$$

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**)

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

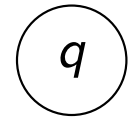
besteht aus:

- einer endlichen Menge Σ, dem **Eingabealphabet**,
- einer endlichen Menge Q, der **Zustandsmenge** (die Elemente aus Q werden Zustände genannt),
- einer Funktion δ von $Q \times \Sigma$ nach Q, der **Übergangsfunktion** (oder Überführungsfunktion),
- 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“).

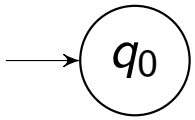
Das Zustandsdiagramm (bzw. der Automatengraph), die grafische Darstellung eines DFA

Grafische Darstellung: Das Zustandsdiagramm

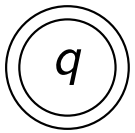
$A = (\Sigma, Q, \delta, q_0, F)$ sei ein DFA.

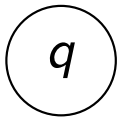
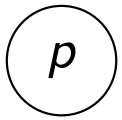
- Für jeden Zustand $q \in Q$ gibt es einen durch  dargestellten Knoten.

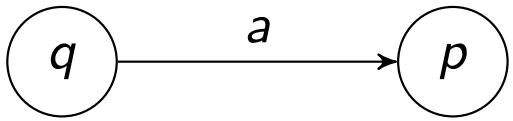
- Der Startzustand q_0 wird durch einen in ihn hinein führenden Pfeil markiert, d.h.:



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

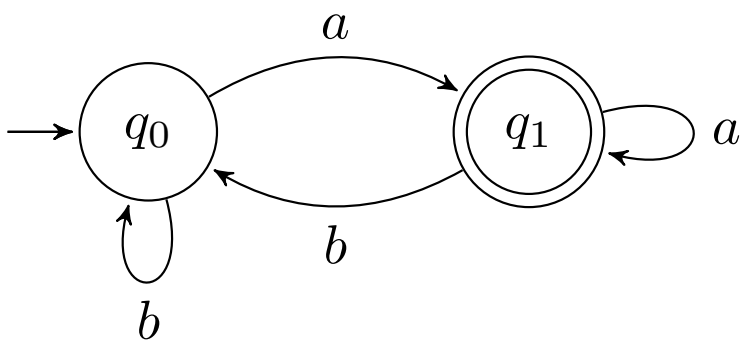


- Seien $p, q \in Q$ Zustände und $a \in \Sigma$ ein Symbol des Alphabets mit $\delta(q, a) = p$. Dann füge einen mit dem Symbol a beschrifteten Pfeil von Knoten  zu Knoten  ein, d.h.:



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 $\delta : Q \times \Sigma \to Q$: siehe Tafel.
4. der Startzustand q_0 und
5. die Menge $F = \{q_1\}$ der akzeptierenden Zustände.

Wie arbeitet ein DFA?
Die erweiterte Übergangsfunktion

Wie arbeitet ein DFA $A = (\Sigma, Q, \delta, q_0, F)$? (1/2)

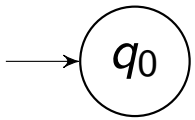
Ein DFA $A = (\Sigma, Q, \delta, q_0, F)$ erhält als Eingabe ein Wort $w \in \Sigma^*$.
(Das Wort repräsentiert eine Folge von „Aktionen“ oder „Bedienoperationen“.)

A wird im Startzustand q_0 gestartet.

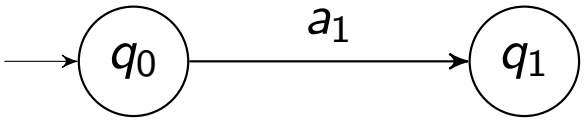
0. Falls w das leere Wort ist, d.h. $w = \varepsilon$, dann verbleibt A im Zustand q_0.
1. Also gelte $w = a_1 \cdots a_n$ mit $n \in \mathbb{N}_{>0}$ und $a_1, \ldots, a_n \in \Sigma$. Der Automat liest den ersten Buchstaben a_1 von w und wechselt in den Zustand

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

In der grafischen Darstellung von A wird der Zustand



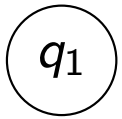
durch die mit a_1 beschriftete Kante verlassen, und q_1 ist der Endknoten dieser Kante, d.h.

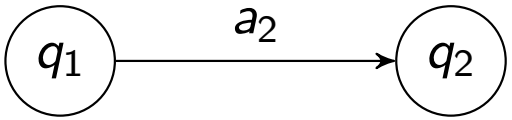


Wie arbeitet ein DFA $A = (\Sigma, Q, \delta, q_0, F)$? (2/2)

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

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

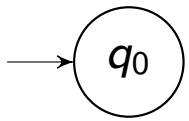
In der grafischen Darstellung von A wird 

durch die mit a_2 beschriftete Kante 

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 $q_1, \ldots, q_n$.
- In der grafischen Darstellung von A entspricht diese Zustandsfolge einem Weg der Länge n, der im Knoten



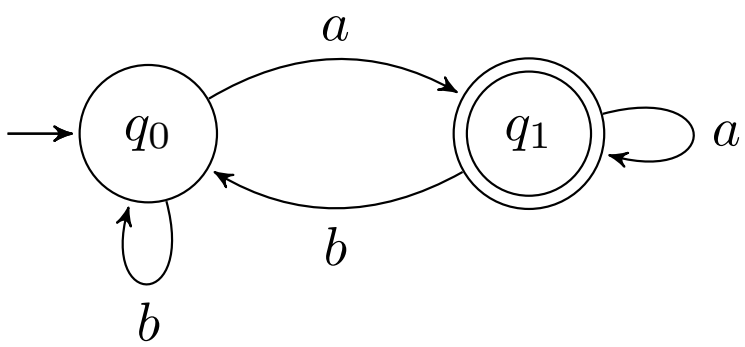
startet und dessen Kanten mit den Buchstaben $a_1, \ldots, a_n$ beschriftet sind.

- 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 $\widehat{\delta}(q, w)$?

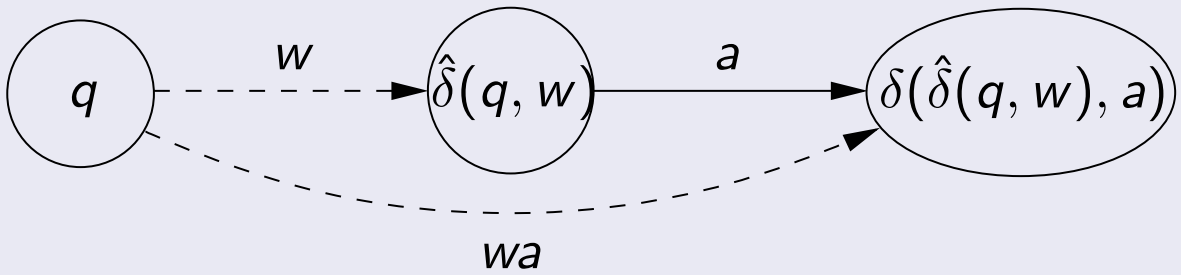
Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein DFA. Eine rekursive Definition der Funktion

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

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

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

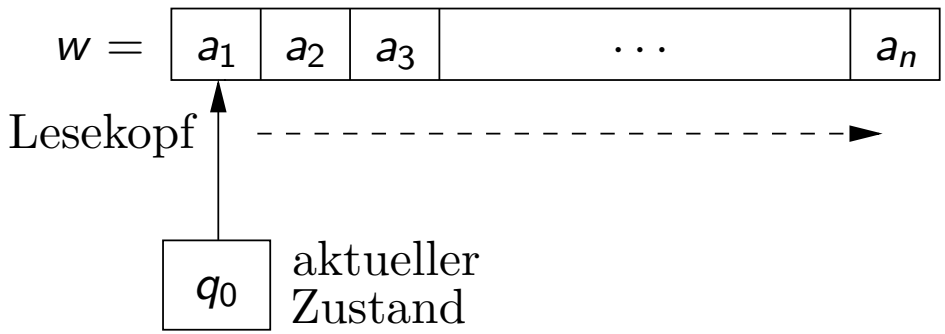
Grafische Darstellung:



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

DFAs: Die Maschinensichtweise

Verarbeitung eines Eingabeworts durch einen DFA A:



Wir können uns einen DFA als eine Maschine vorstellen, die

* ihre Eingabe $w = a_1 a_2 a_3 \cdots a_n$ 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

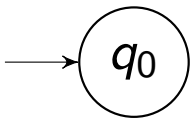
$$\widehat{\delta}(q_0, w) \in F,$$

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

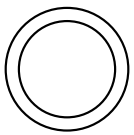
Wie „übersetzt“ sich

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

in der grafischen Darstellung von A? Es gibt einen in



startenden Weg der Länge n,

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

Die akzeptierte Sprache $L(A)$

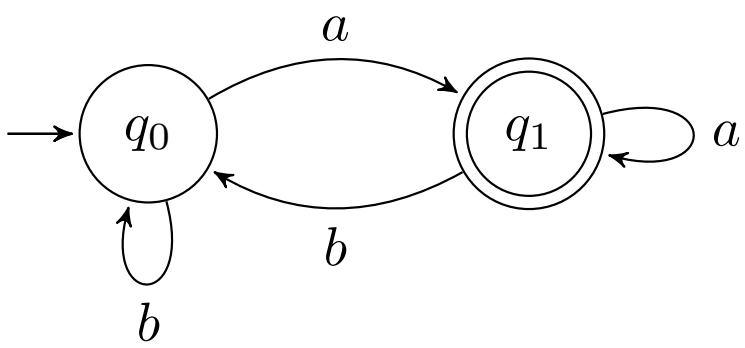
Die von einem DFA $A = (\Sigma, Q, \delta, q_0, F)$ akzeptierte Sprache **L(A)** ist

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

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

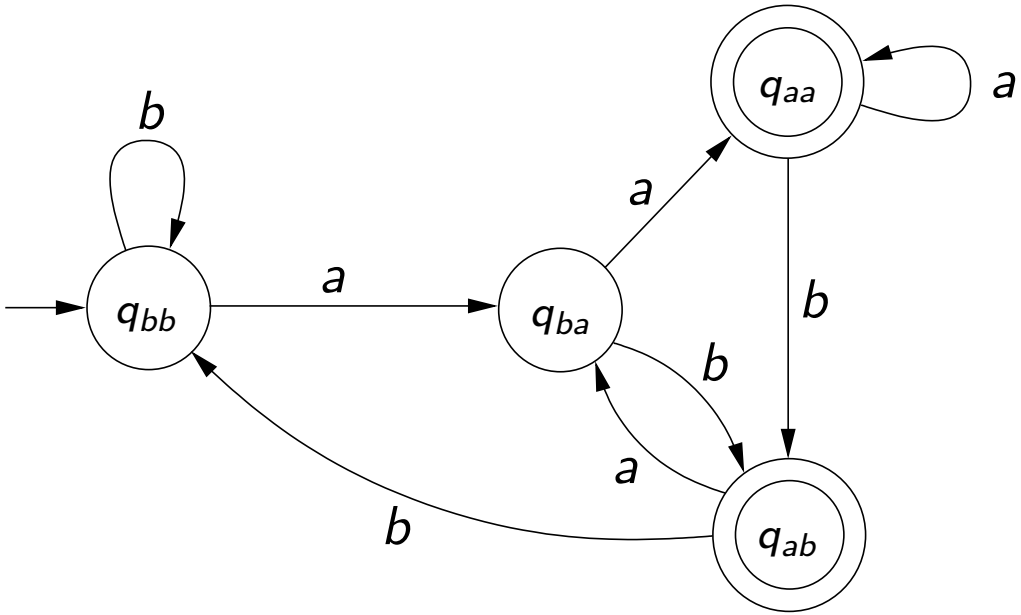


A_1 akzeptiert die Sprache

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

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

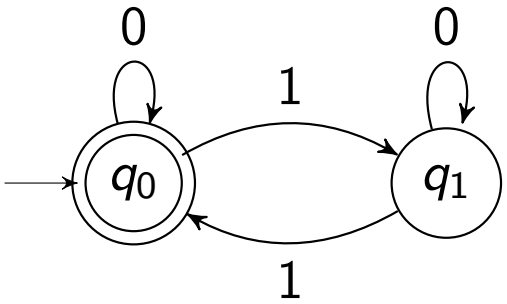
Bei der Speicherung von Daten auf einem Speichermedium eines Computers werden Informationen durch binäre Worte über dem Alphabet $\{0, 1\}$ **kodiert**.

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

$$\mathbf{L(A) = L(B)}.$$

(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 „**aquivalent**“ 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

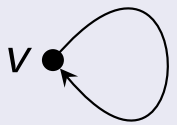
sind äquivalente Konzepte!

Wichtige Eigenschaften 2-stelliger Relationen

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

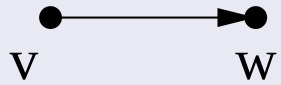
(a) E heißt **reflexiv**, falls für alle $v \in V$ gilt:

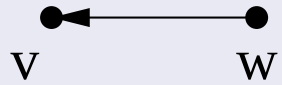
$$(v, v) \in E. \qquad \text{(Skizze: } v \text{)}$$



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

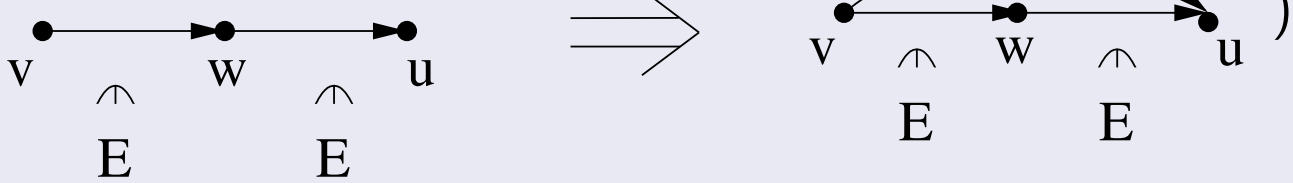
$$\text{Wenn } (v, w) \in E, \text{ dann auch } (w, v) \in E.$$

Zur Kante 

gibt es auch die "Rückwärtskante" 

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

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

(Skizze: 

)

Äquivalenzrelationen

(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** 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.

Äquivalenzrelationen: Beispiele (1/2)

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

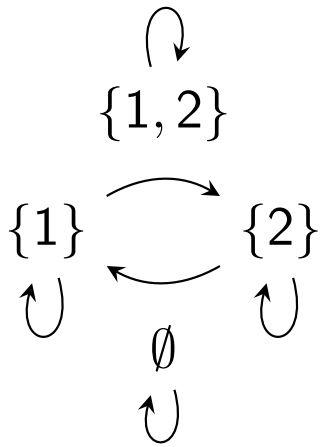
$$E := \{ (m, m) : m \in M \}$$

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* $M = \{1, 2\}$*:*



Äquivalenzrelationen: Beispiele (2/2)

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

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

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.

Der Index

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

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

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: $p \equiv_A q :\Longleftrightarrow$

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

(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 wichtigen Fragen

1. Ist die Verschmelzungsrelation $\equiv_A$ eine Äquivalenzrelation?
2. Wir möchten alle äquivalenten Zustände zu einem einzigen Zustand **verschmelzen**.
   - Dürfen wir das?
   - Gilt: „Index der Verschmelzungsrelation $\equiv_A$“ = „Zustandszahl eines minimalen Automaten“? (Wenn ja, haben wir einen minimalen Automaten gefunden!)

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 $\mathbf{A'}$ 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 $\equiv_A$

Zeugen (1/3)

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

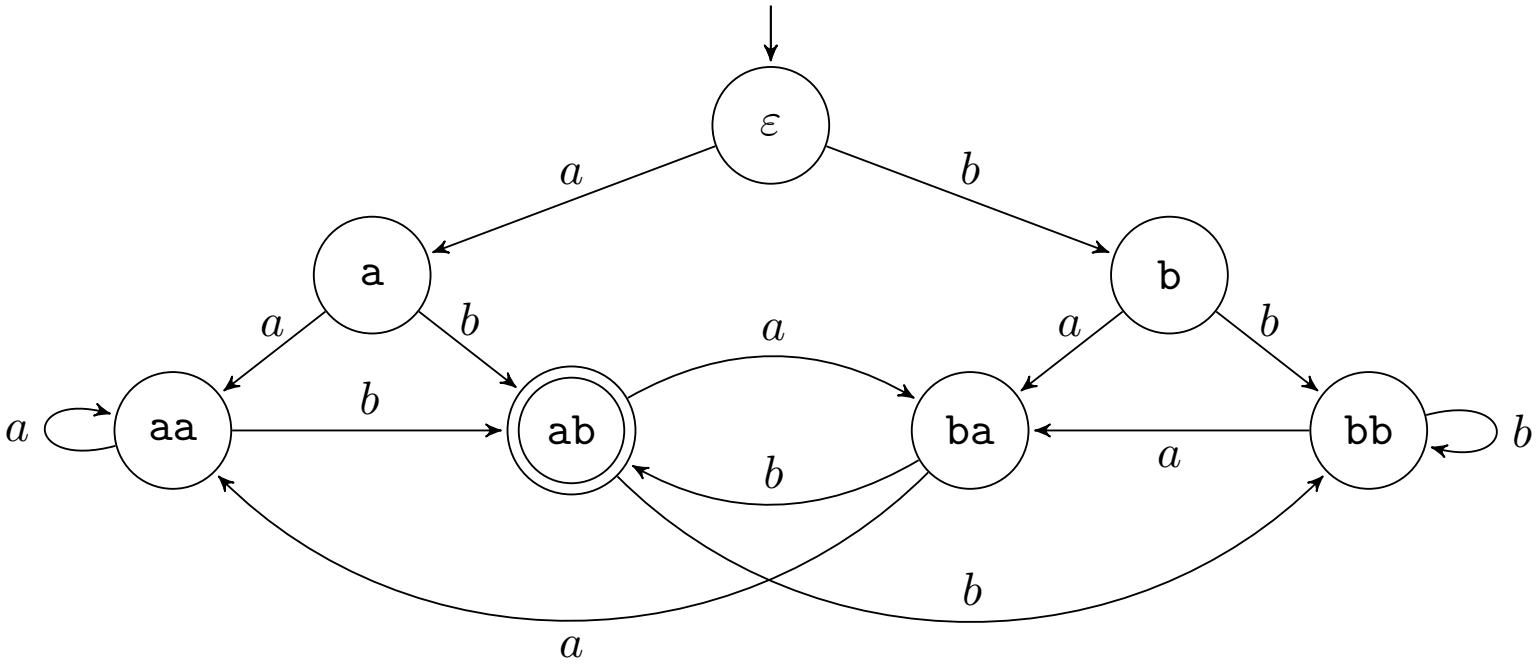
(a) Das Wort $w \in \Sigma^*$ ist **Zeuge** für die Inäquivalenz von p und q, wenn

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

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

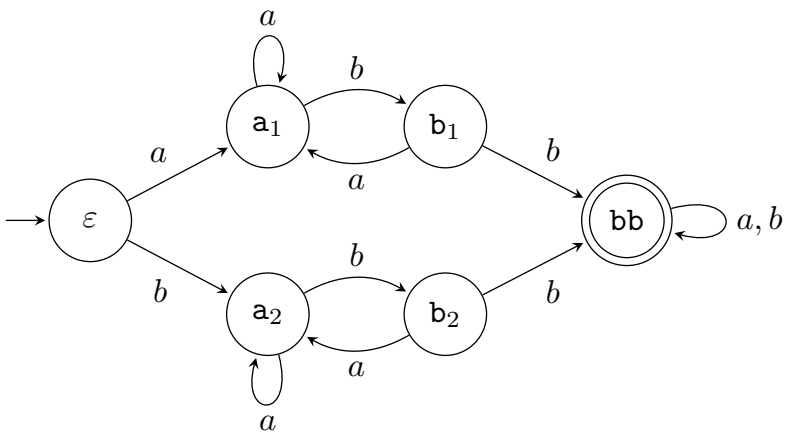
(b) Es ist $p \not\equiv_A q$ genau dann, wenn es einen Zeugen für die Inäquivalenz von p und q gibt, bzw wenn es ein Wort gibt, das p und q trennt.

Zeugen (2/3)



- Finde einen Zeugen für die Inäquivalenz von ab und ba.
- Welcher Zeuge trennt ε und a ?
- Gibt es Zeugen, die die Zustände a und aa trennen?

Zeugen (3/3)



- Finde einen Zeugen für die Inäquivalenz von b_1 und bb.
- Welcher Zeuge trennt ε 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 $r \not\equiv_A s$, gibt es einen **Zeugen** w mit

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

   - Das Wort aw **trennt** p und $q \Longrightarrow aw$ 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 $p \not\equiv_A q$ 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
   - $\big(\delta(p, \varepsilon) \in F$ und $\delta(q, \varepsilon) \notin F\big)$ oder $\big(\delta(p, \varepsilon) \notin F$ und $\delta(q, \varepsilon) \in F\big)$,
   - bzw. $\big(p \in F$ und $q \notin F\big)$ oder $\big(p \notin F$ und $q \in F\big)$.

   Aber dann haben wir $\{p, q\}$ in die Menge M_0 eingefügt. ↯
2. Wenn $w = au$ für den Buchstaben $a \in \Sigma$, dann ist
   - $\big(\widehat{\delta}(p, au) \in F$ und $\widehat{\delta}(q, au) \notin F\big)$ oder $\big(\widehat{\delta}(p, au) \notin F$ und $\widehat{\delta}(q, au) \in F\big)$,
   - bzw. $\big(\widehat{\delta}(\delta(p, a), u) \in F$ und $\widehat{\delta}(\delta(q, a), u) \notin F\big)$ oder $\big(\widehat{\delta}(\delta(p, a), u) \notin F$ und $\widehat{\delta}(\delta(q, a), u) \in F\big)$.

   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 δ' 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 $p \not\equiv_A q$:

1. Zuerst bestimme $M_0 := \Big\{ \{p, q\} \ : \ p \in F, \ q \in Q \setminus F \Big\}$; Setze $i := 0$
2. Wiederhole
3. Für alle Paarmengen $\{p, q\}$ mit $p \neq q$, $\{p, q\} \notin (M_0 \cup \cdots \cup M_i)$ und für alle $a \in \Sigma$ tue folgendes:
4. Falls $\{\delta(p, a), \delta(q, a)\} \in M_i$, füge $\{p, q\}$ zur Menge M_{i+1} hinzu.
5. $i := i + 1$
6. bis $M_i = \emptyset$
7. **Ausgabe:** $M := M_0 \cup \cdots \cup M_{i-1}$.

Schritt 3: Konstruiere $A' := (Q', \Sigma, \delta', q_0', F')$:

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

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

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

Ist der Algorithmus korrekt und effizient?

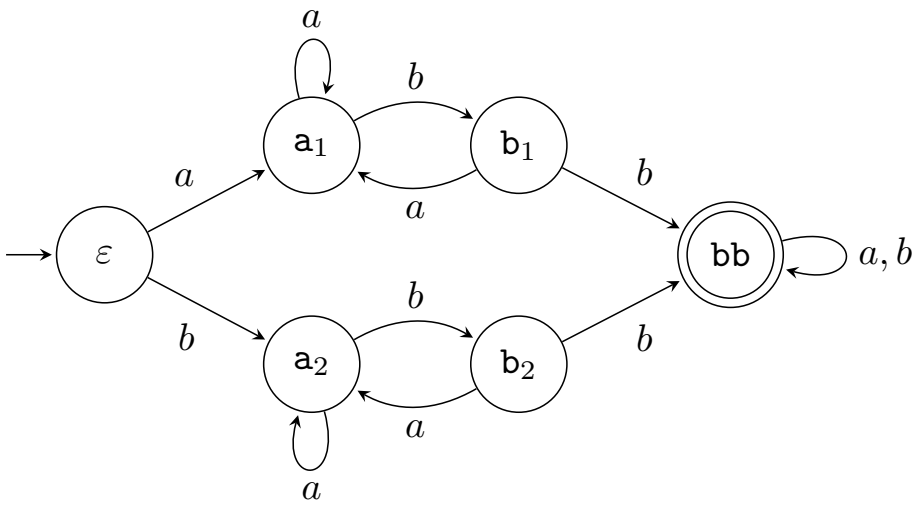
In jeder Iteration in Schritt 2 wird mindestens eine neue Paarmenge markiert:
Es gibt also höchstens $|Q|^2$ Iterationen und der Algorithmus ist „**effizient**“.

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 ein zweites Beispiel durch.

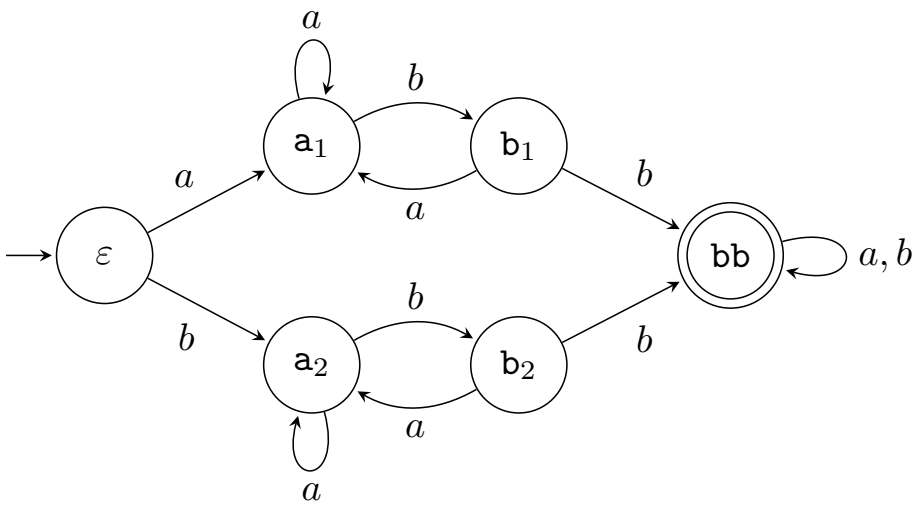
Zustandsminimierung: Die Mengen M_0, M_1, M_2



a_1					
a_2					
b_1					
b_2					
bb					
	ε	a_1	a_2	b_1	b_2

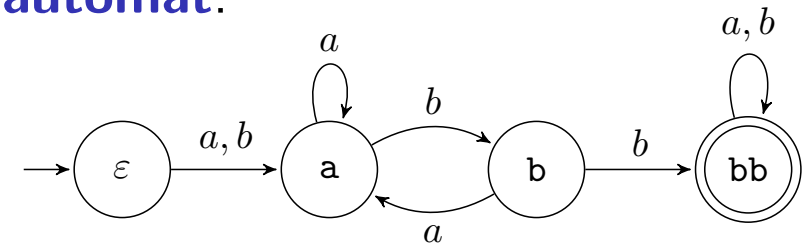
a_1				
a_2				
b_1				
b_2				

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 $M_0, \ldots, M_i$ bestimmt und die „entsprechenden“ Positionen in der Tabelle markiert.

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

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

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

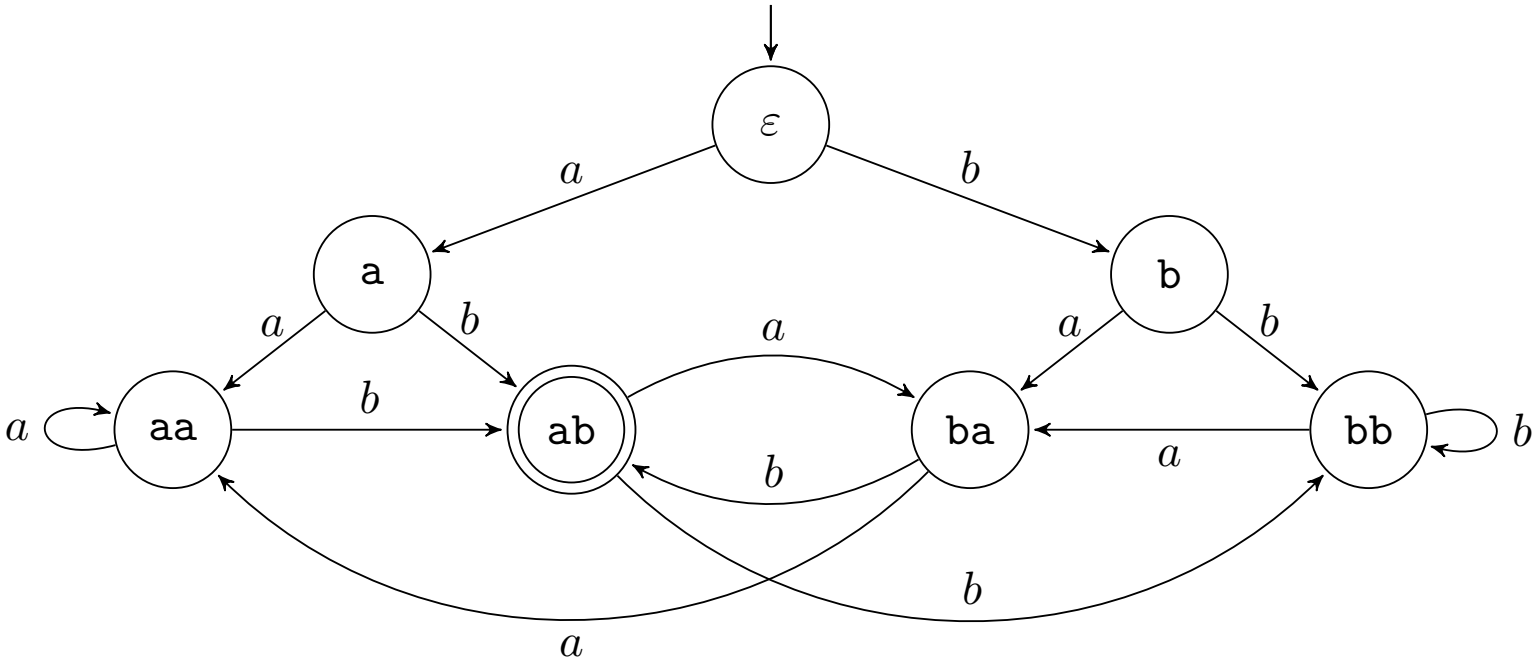
- Wir arbeiten mit dem Alphabet $\Sigma = \{a, b\}$.
- Speichere die beiden zuletzt gelesenen Buchstaben im aktuellen Zustand.
- Wir benutzen deshalb die Zustände

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

  - 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 $\Longrightarrow$ Die Menge M_0 hat die Form

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

a						
b						
aa						
ab	M_0	M_0	M_0	M_0		
ba					M_0	
bb					M_0	
	ε	a	b	aa	ab	ba

Was haben wir gelernt?

(a) $\{ab\}$ ist eine Klasse der Verschmelzungsrelation,

(b) die restlichen Klassen sind disjunkte Teilmengen von $\{\varepsilon, a, b, aa, ba, bb\}$.

Wie sieht M_1 aus?

Zustandsminimierung: Die Menge M_1

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

a						
b						
aa						
ab	M_0	M_0	M_0	M_0		
ba					M_0	
bb					M_0	
	ε	a	b	aa	ab	ba
a	M_1					
b		M_1				
aa						
ab	M_0	M_0	M_0	M_0		
ba					M_0	
bb		M_1			M_0	
	ε	a	b	aa	ab	ba
a	M_1					
b		M_1				

Was haben wir gelernt?

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
	ε	a	b	aa	ab	ba

1. Wir haben jeden der Zustände a, aa, ba von jedem der Zustände ε, b, bb getrennt.
2. Die weiteren Äquivalenzklassen der Verschmelungsrelation sind disjunkte Teilmengen
   - entweder von $\{a, aa, ba\}$
   - oder von $\{\varepsilon, b, bb\}$.

Lassen sich die Zustände in $\{a, aa, ba\}$ (bzw. in $\{\varepsilon, b, bb\}$) voneinander trennen?

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
	ε	a	b	aa	ab	ba

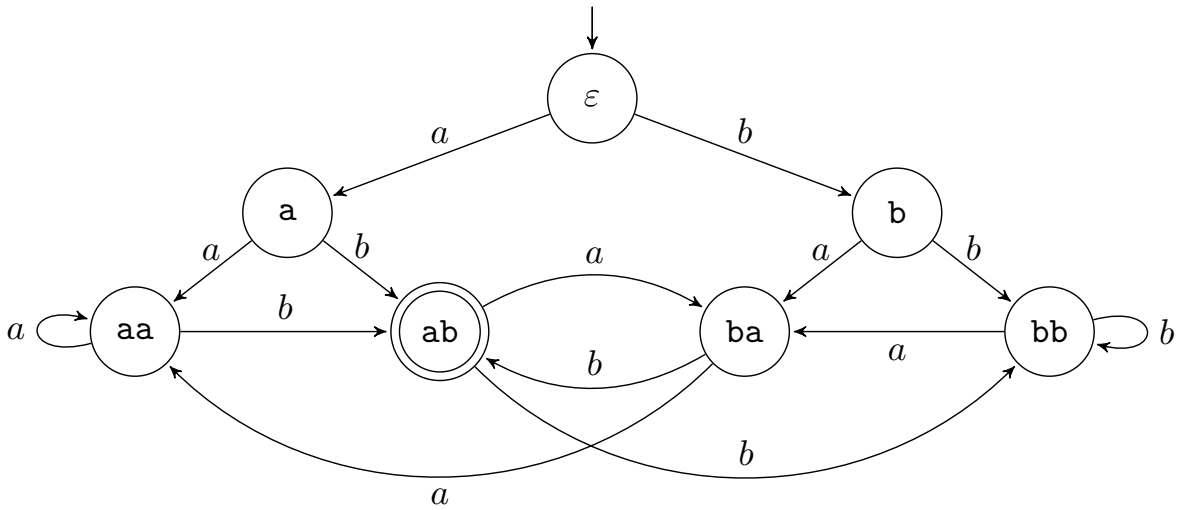
1. Die Zustände a, aa, ba können nicht voneinander getrennt werden, da sie alle unter a auf den Zustand aa und unter b auf den Zustand ab führen.
2. b und bb lassen sich ebenfalls nicht mehr trennen. Beide führen unter a auf ba und unter b auf bb.
3. Aber auch ε führt unter a auf die Klasse $\{a, aa,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

Es ist $M_2 = \emptyset$, denn ε lässt sich nicht von b, bb trennen.

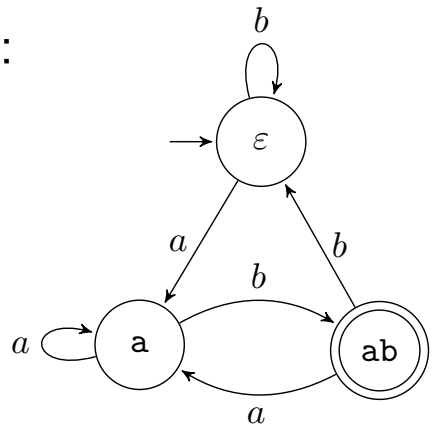
Zustandsminimierung: Der Äquivalenzklassenautomat A'

(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?

1. Tafel: Der ursprüngliche Automat $A = (\Sigma, Q, \delta, q_0, F)$ und sein Äquivalenzklassenautomat A' sind äquivalent.
   - Jede Klasse der Verschmelzungsrelation hat entweder nur akzeptierende Zustände oder gar keine akzeptierenden Zustände. ✓
   - Wenn $p \equiv_A q$ und $a \in \Sigma$, dann gilt $\delta(p, a) \equiv_A \delta(q, a)$. ✓
   - Für alle Worte $w \in \Sigma^*$ gilt: Der von A' nach Lesen von w im Zustand q'_0 erreichte Zustand ist die Äquivalenzklasse von $\hat{\delta}(q_0, w)$. ✓

✓ 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 $\widehat{\delta}(q_0, u) = \widehat{\delta}(q_0, v)$, dann

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

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

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

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

(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

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

(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 $\equiv_A$.

Die Nerode-Relation: Beispiele

Wie sehen die Äquivalenzklassen der Nerode-Relation in Beispielen aus? Insbesondere, wie groß ist 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 hat eine gerade Anzahl von Einsen $\}$

(∗) Es ist $\varepsilon \not\equiv_P 1$ mit dem Zeugen ε.

(∗) Sei u ein beliebiges Wort mit gerade vielen Einsen und v ein beliebiges Wort mit ungerade vielen Einsen. Dann gilt **für alle** $w \in \Sigma^*$

$$\Big(\varepsilon w \in P \iff uw \in P\Big) \text{ und } \Big(1w \in P \iff vw \in P\Big).$$

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

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

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

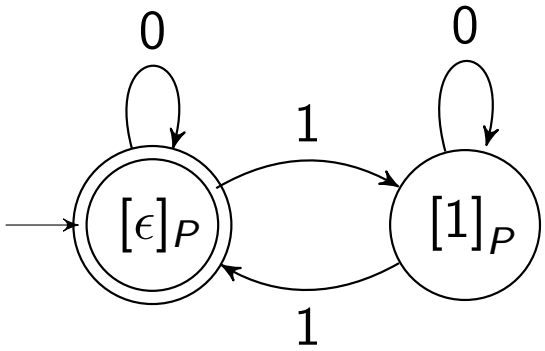
Baue aus diesen beiden Äquivalenzklassen einen DFA N_P, der P akzeptiert.

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 muss die Äquivalenzklasse von $\epsilon 0 = 0$ sein und das ist die Klasse $[\epsilon]_P$.
   - 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 und das bleibt die Klasse $[1]_P$.
   - Lesen wir eine 1, ist die Klasse von 11, also die Klasse $[\epsilon]_P$ unser Ziel.

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



Worauf ist beim Bau des Nerode-Automaten zu achten?

Sei L eine beliebige Sprache über dem Alphabet Σ 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 $\lightning$

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

1. Wähle einen Vertreter v_K für jede Nerode-Klasse K,
2. Für alle Nerode-Klassen K und alle Buchstaben $a \in \Sigma$
   - bestimme den Übergang $\delta_{N_L}(v_K, a)$ und
   - bestimme den Vertreter v_{K^*} mit $v_{K^*} \equiv_{N_L} \delta_{N_L}(v_K, a)$.

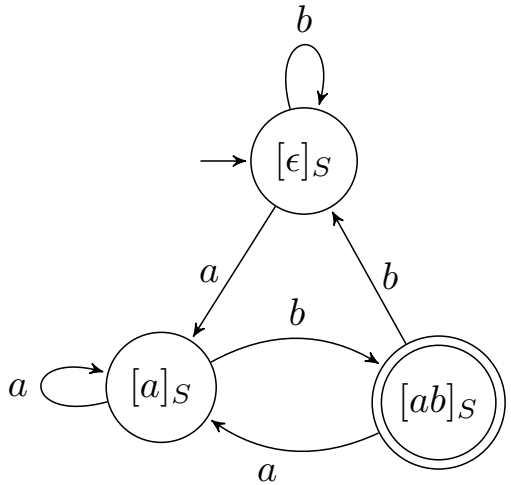
Der Nerode-Automat N_S der Suffixsprache $S = \{a, b\}^* \cdot ab$

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

Der **Nerode-Automat** N_S:



Der Nerode-Automat (1/2)

Sei L eine Sprache über dem Alphabet Σ. 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.

Der Nerode-Automat (2/2)

Sei L eine Sprache über dem Alphabet Σ.

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

$$\widehat{\delta}([\varepsilon]_L, w) = [\mathrm{w}]_L.$$

(b) N_L akzeptiert die Sprache L mit Index(L) vielen Zuständen.

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

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

Induktionsschritt: Für $w = ua$ ist

$$\widehat{\delta}(q_0, w) = \widehat{\delta}(q_0, ua) = (\delta(\widehat{\delta}(q_0, u), a) = \delta([\mathrm{u}]_L, a) := [\mathrm{ua}]_L. \checkmark$$

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 und $[u]_L \in F$ gilt genau dann wenn $u \in L$.

Der Satz von Myhill-Nerode I

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

(a) 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.

- $u_1, \ldots, u_k$ 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

  $\Longrightarrow \widehat{\delta}(q_0, u_i) = \widehat{\delta}(q_0, u_j)$

  $\Longrightarrow \widehat{\delta}(q_0, u_i w) = \widehat{\delta}(q_0, u_j w)$ für alle Worte $w \in \Sigma^*$.

  $\Longrightarrow u_i w \in L \Longleftrightarrow u_j w \in L$ für alle Worte $w \in \Sigma^*$ und $u_i \equiv_L u_j$ ↯.
- Es ist $|Q| \geqslant \text{Index}(L)$, aber der Nerode-Automat N_L akzeptiert L mit genau Index(L) vielen Zuständen.

(b) 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 $L(A) = L$.
$\mathbf{A'} = (Q', \Sigma, \delta', q_0', F')$ sei sein Äquivalenzklassenautomat.

Angenommen, Worte $u, v \in \Sigma^*$ führen in $\mathbf{A'}$ 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: $\left(\widehat{\delta}(p, w) \in F \wedge \widehat{\delta}(q, w) \notin F\right) \vee \left(\widehat{\delta}(p, w) \notin F \wedge \widehat{\delta}(q, w) \in F\right)$, bzw.
   - $\left(\widehat{\delta}(q_0, uw) \in F \wedge \widehat{\delta}(q_0, vw) \notin F\right) \vee \left(\widehat{\delta}(q_0, uw) \notin F \wedge \widehat{\delta}(q_0, vw) \in F\right)$, bzw.
   - $\Big(uw \in L \wedge vw \notin L\Big) \vee \Big(uw \notin L \wedge vw \in L\Big)$.
3. Wenn u und v in $\mathbf{A'}$ zu verschiedenen Zuständen führen, dann folgt $u \not\equiv_L v$.
4. Die Zustandszahl von A' ist höchstens so groß wie Index(L)

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

Der Äquivalenzklassenautomat $\mathbf{A'}$ ist ein minimaler DFA für L(A).

Reguläre Sprachen

Reguläre Sprachen

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

$$L = L(A).$$

(a) Gibt es Teilmengen von Σ^*, die **keine** regulären Sprachen sind?

(b) Ist $L = \{ a^n b^n \ : \ n \in \mathbb{N} \}$ eine reguläre Sprache?

(c) Sei Σ ein beliebiges Alphabet und sei $w \in \Sigma^*$ ein Wort über Σ. Ist

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

eine reguläre Sprache?

Der Satz von Myhill-Nerode II

Wann ist eine Sprache regulär?

Satz von Myhill-Nerode II

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

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

- A hat endlich viele Zustände, aber mindestens Index(L) Zustände.
- Also ist Index(L) endlich.

$\Longleftarrow$ Index(L) sei endlich.

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

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

Zeige: 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_ℓ:

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

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_ℓ:

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

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

DFAs können sich nur beschränkt viele Dinge merken.

Weitere nicht-reguläre Sprachen

Keine der folgenden Sprachen ist regulär.

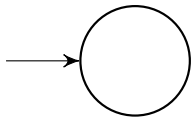
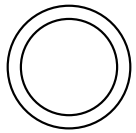
- $L_1 = \{ a^n b^m \ : \ n, m \in \mathbb{N},\ n \leqslant m \}$.
- $L_2 = \{ a^n b^m c^{n+m} \ : \ n, m \in \mathbb{N} \}$.
- $L_3 = \{ a^{n^2} \ : \ n \in \mathbb{N} \}$.
- $L_4 = \{ w \in \{a, b\}^* \ : \ w$ ist ein Palindrom $\}$:

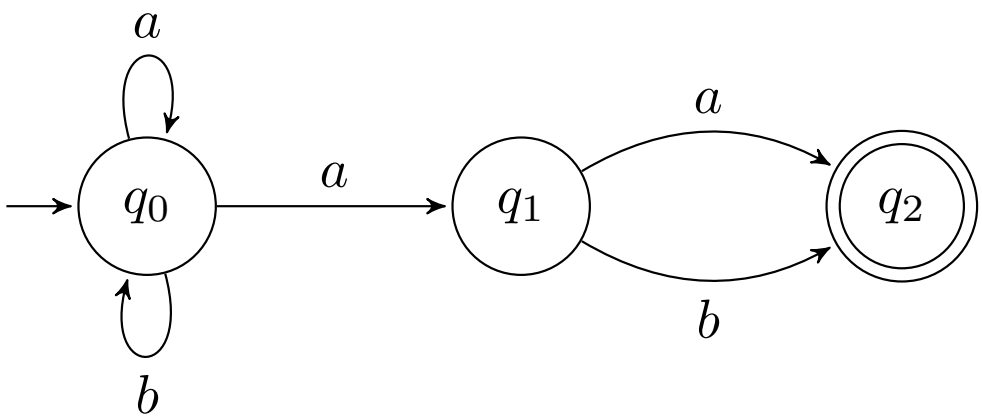
Zeige jeweils, dass der Index unendlich ist.

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

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

NFAs: Die formale Definition

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

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

besteht aus:

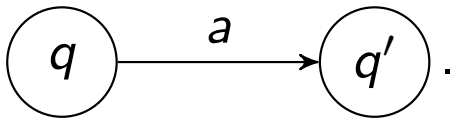
- einer endlichen Menge Σ, dem Eingabealphabet,
- einer endlichen Menge Q, der Zustandsmenge,
- dem Startzustand $q_0 \in Q$,
- einer Menge $F \subseteq Q$ von Endzuständen bzw. akzeptierenden Zuständen und
- einer Funktion $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$, der Übergangsfunktion,
  - *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.



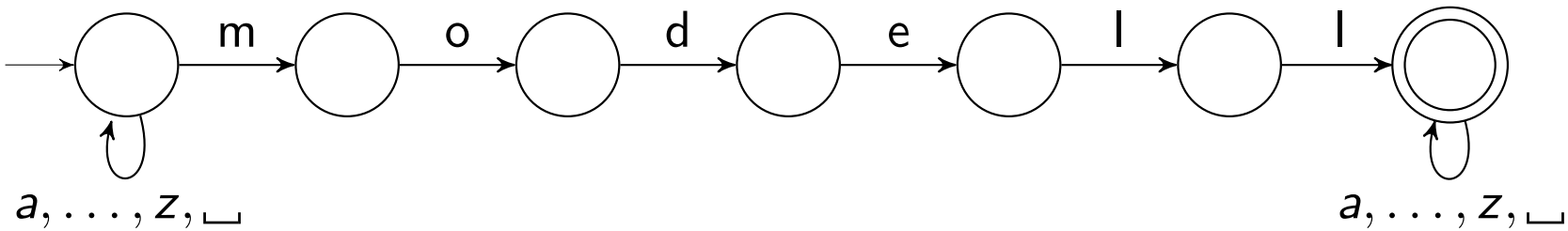
Stichwortsuche mit NFAs (1/2)

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:

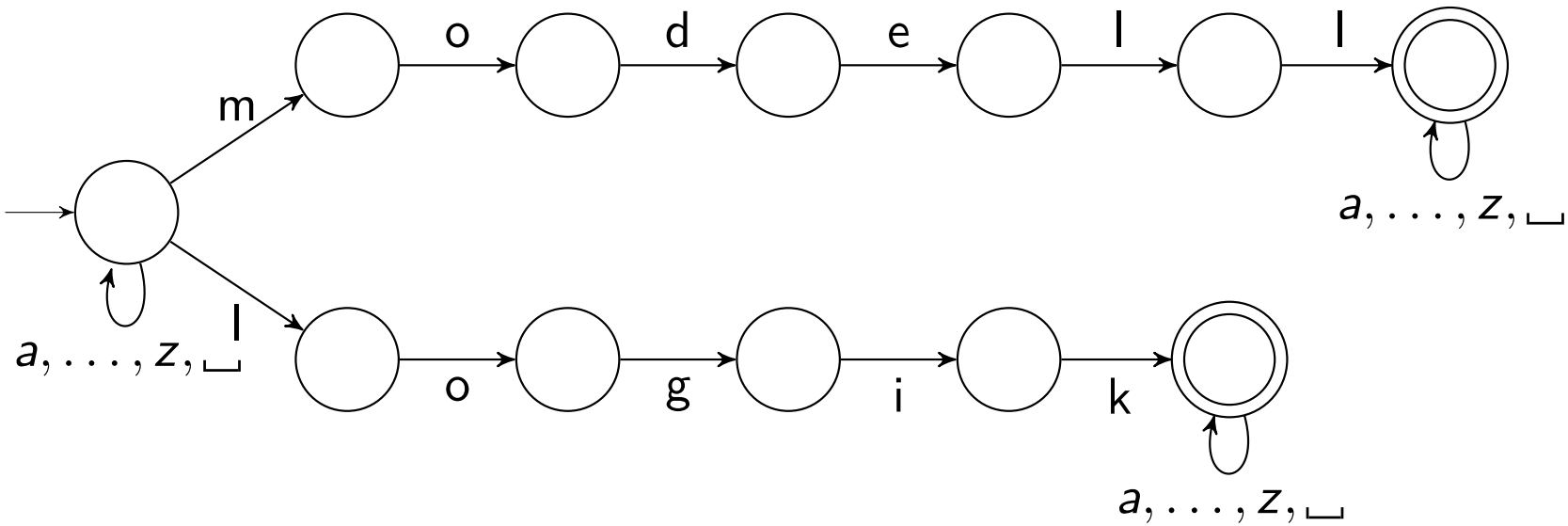


Stichwortsuche mit NFAs (2/2)

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

> *Kommt mindestens eins der Stichworte „modell“ bzw. „logik“ im Eingabetext vor?*

Graphische Darstellung eines NFAs, der dies bewerkstelligt:

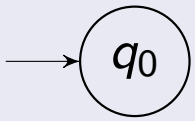


Die von einem NFA akzeptierte Sprache

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 $a_1 \ldots a_n$ 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

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

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 relativ wenigen Zuständen.

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

Die Potenzmengenkonstruktion (1/2)

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

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

Die Potenzmengenkonstruktion (2/2)

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. dass gilt:

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

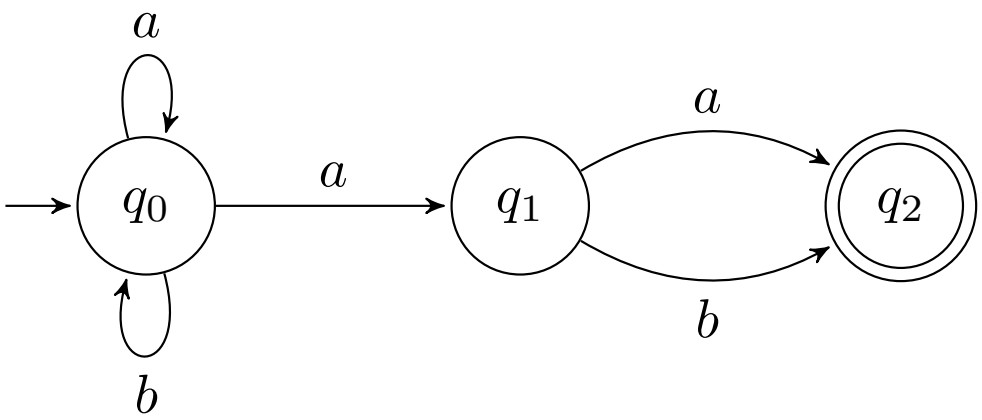
Und wie, bitte schön, sollen wir das zeigen?

- 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.

Die Potenzmengenkonstruktion: Ein Beispiel (1/2)

Wie führt man die Potenzmengenkonstruktion für den NFA

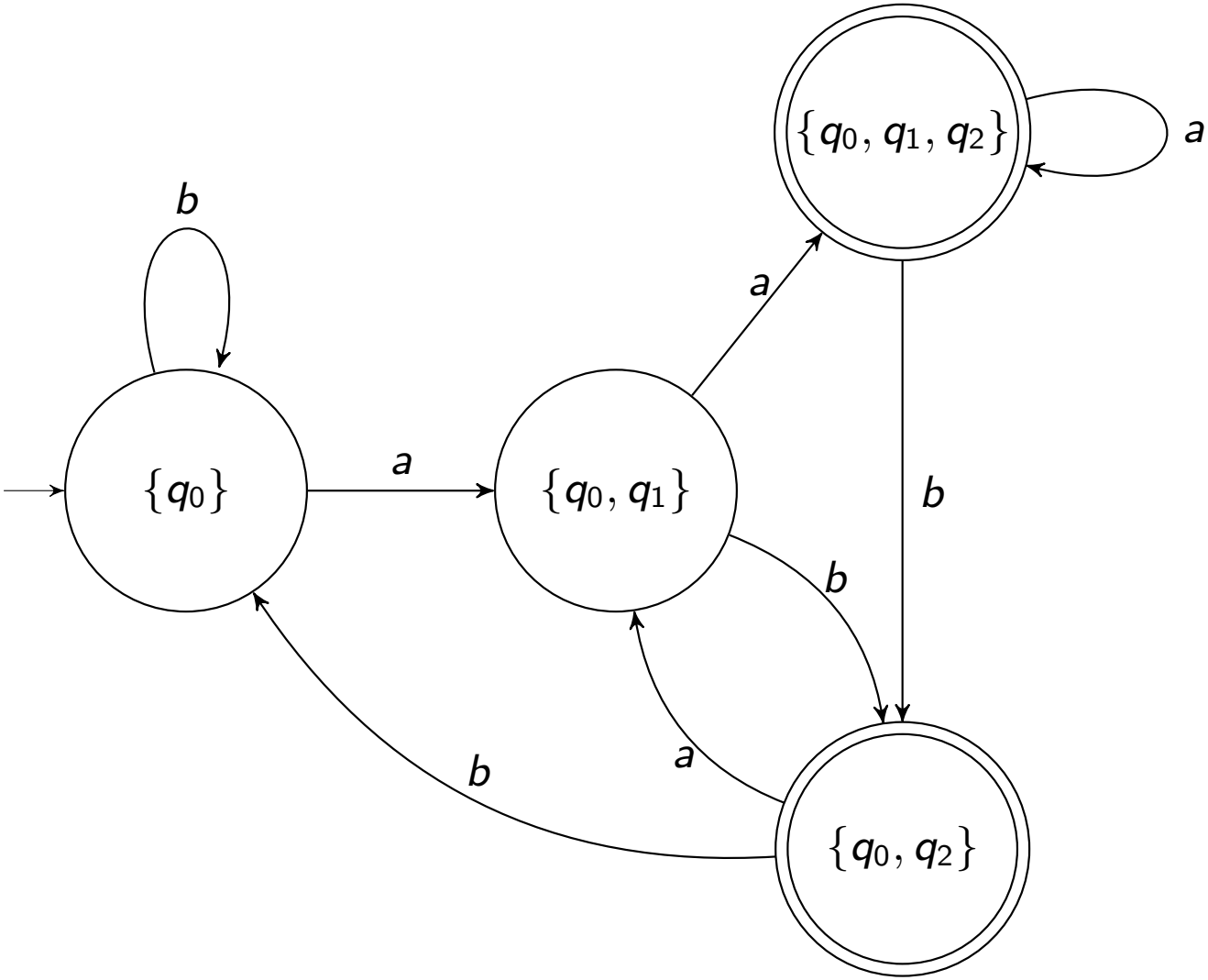


aus?

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.

Die Potenzmengenkonstruktion: Ein Beispiel (2/2)



Reguläre Ausdrücke

Reguläre „Muster“

Sei Σ ein Alphabet.

1. Die Menge aller Worte über Σ haben wir beschrieben mit

$$\Sigma^*.$$

2. Sei u ein Wort über Σ. Dann wird die Menge aller Worte über Σ, die u als Teilwort besitzen, beschrieben durch

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

3. Die Menge aller Worte über Σ, deren vorletzter Buchstabe ein a ist, wird beschrieben durch:

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

Reguläre Ausdrücke: Eine rekursive Definition der Syntax

Sei Σ ein endliches Alphabet.

Die Menge aller **regulären Ausdrücke** über Σ ist rekursiv wie folgt definiert:

Basisregeln:

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

Rekursive Regeln:

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

Reguläre Ausdrücke: Syntax und Semantik

Wir haben gerade **formal definiert**,

„was ein regulärer Ausdruck R ist“

Aber was

„**bedeutet**“ R?

R sollte für eine Sprache stehen!

Reguläre Ausdrücke: Eine rekursive Definition der Semantik

Sei Σ ein endliches Alphabet.

Jeder reguläre Ausdruck R über Σ **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 Σ, so ist

$$L(R^*) := \{\varepsilon\} \cup \{\, w_1 \cdots w_k : k \in \mathbb{N}_{>0},\, w_1 \in L(R), \ldots, w_k \in L(R) \,\}.$$

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

Reguläre Ausdrücke: Vereinfachte Schreibweise

Zur vereinfachten Schreibweise und besseren Lesbarkeit regulärer Ausdrücke:

- Den „Punkt“ bei der Konkatenation

$$(R \cdot S)$$

  darf man weglassen.
- Bei Ketten gleichartiger Operatoren verzichten wir auf Klammern:

$$(R_1 \mid R_2 \mid R_3 \mid R_4) \quad \text{statt} \quad \Big(((R_1 \mid R_2) \mid R_3) \mid R_4\Big) \text{ und}$$

$$(R_1 R_2 R_3 R_4) \quad \text{statt} \quad \Big(((R_1 R_2) R_3) R_4\Big).$$

- „Präzedenzregeln“: * bindet stärker als $\cdot$
  $\cdot$ bindet stärker als $\mid$
- Äußere Klammern, die einen regulären Ausdruck umschließen, dürfen weggelassen werden.
- Zur besseren Lesbarkeit dürfen zusätzliche Klammern benutzt werden.

Reguläre Ausdrücke: Beispiele (1/3)

(a) $a \mid bc^*$ ist eine verkürzte Schreibweise für $(a \mid (b \cdot c^*))$.
Die von diesem regulären Ausdruck beschriebene Sprache $L = L(a \mid bc^*)$ ist

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

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

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

$$(a \mid b \mid c)^* abb\, (a \mid b \mid c)^*.$$

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

$$(a \mid b \mid c)^* a\, (\varepsilon \mid a \mid b \mid c).$$

Reguläre Ausdrücke: Beispiele (2/3)

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

$$\textit{Vorwahl}/\textit{Nummer}$$

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

$$0\ (1|2|\cdots|9)\ (0|1|\cdots|9)^*\ /\ (1|2|\cdots 9)\ (0|1|\cdots 9)^*$$

definiert die gewünschte Sprache.

Reguläre Ausdrücke: Beispiele (3/3)

Der Ausdruck:

$$R := \left(\varepsilon \mid 069/\right)\ \ 798\ \ \left(\varepsilon \mid \text{-}\right)\ \ \left(\ 0\ \ \mid (1|2|\cdots|9)\,(0|1|\cdots|9)^*\ \right)$$

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

- ? 069/798-0
- ? 7980
- ? 069/798-028551.

Reguläre Sprachen

Reguläre Sprachen und reguläre Ausdrcke

Jeder reguläre Ausdruck R definiert eine reguläre Sprache.

Beweis durch Induktion über den Aufbau von R: Siehe Tafel.

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!

Reguläre Sprachen: Ein Ausblick

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!