CoNe
Freiburg

Informatik III

2.1 Endliche Automaten

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik

Wintersemester 2007/08

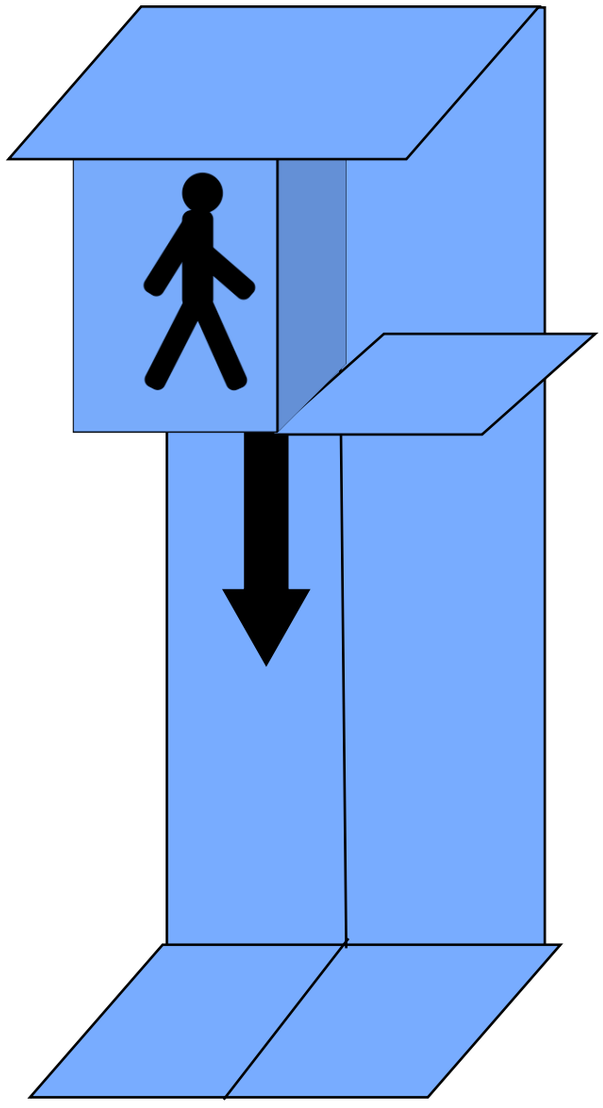
Formale Sprachen und
Endliche Automaten

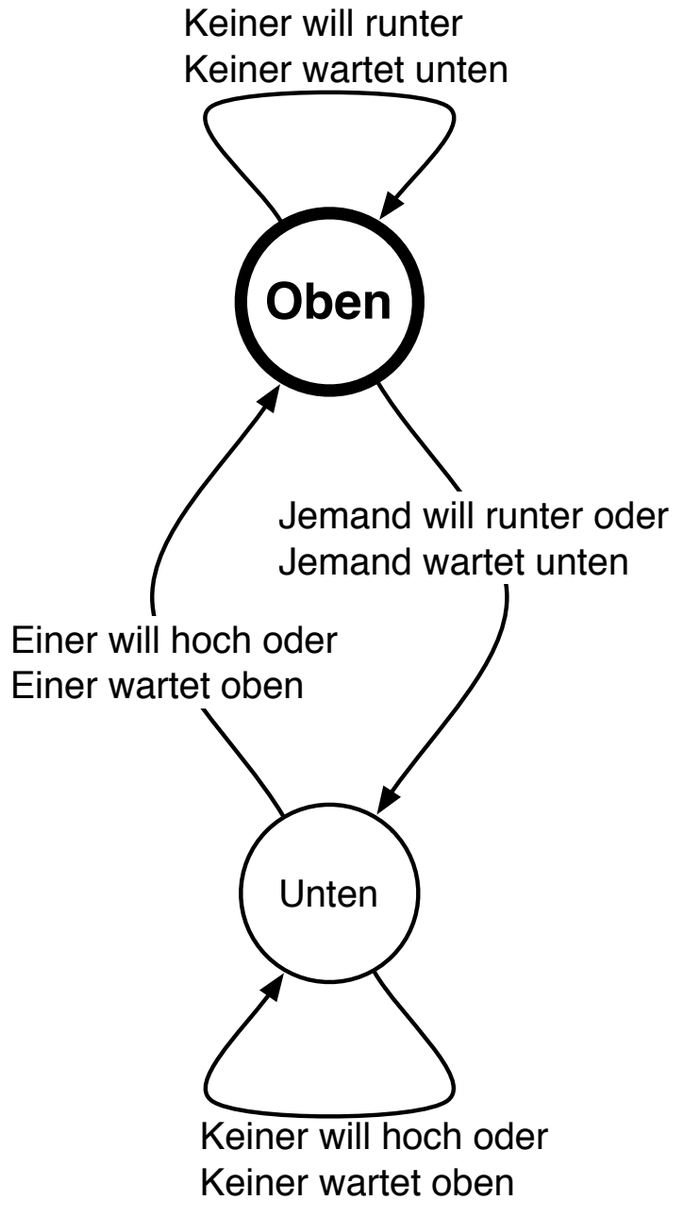
Ein einfacher Automat

2

Ein einfacher Automat

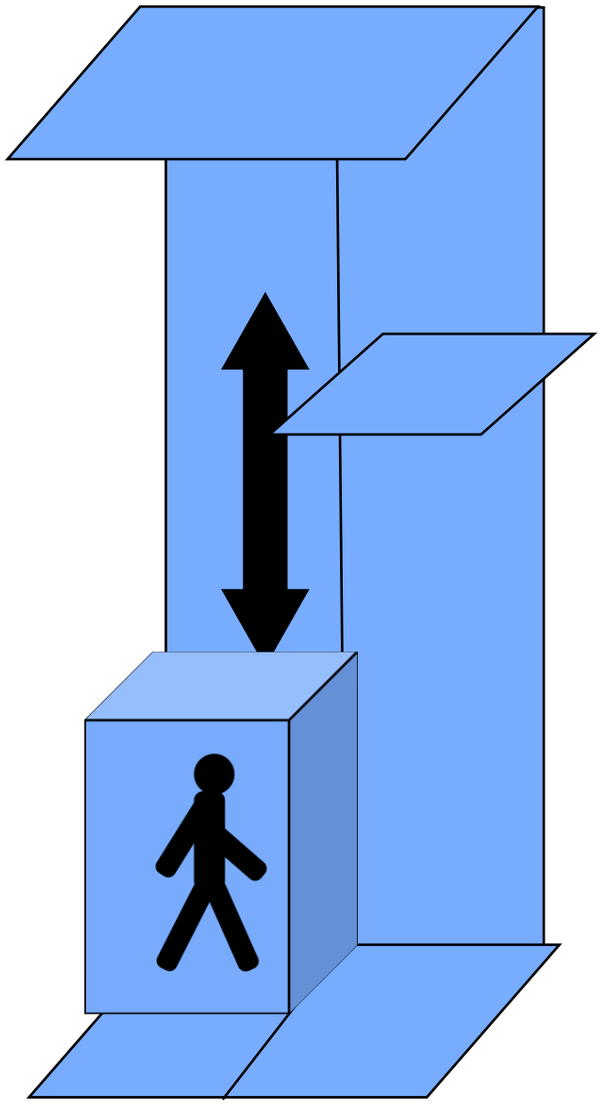
- **Lift mit zwei Stockwerken**
  - Eingaben:
    - Im Lift will jemand nach oben
    - Im Lift will jemand nach unten
    - Unten Bedarf
    - Oben Bedarf
  - Zwei Zustände:
    - **Lift oben**
    - Lift unten

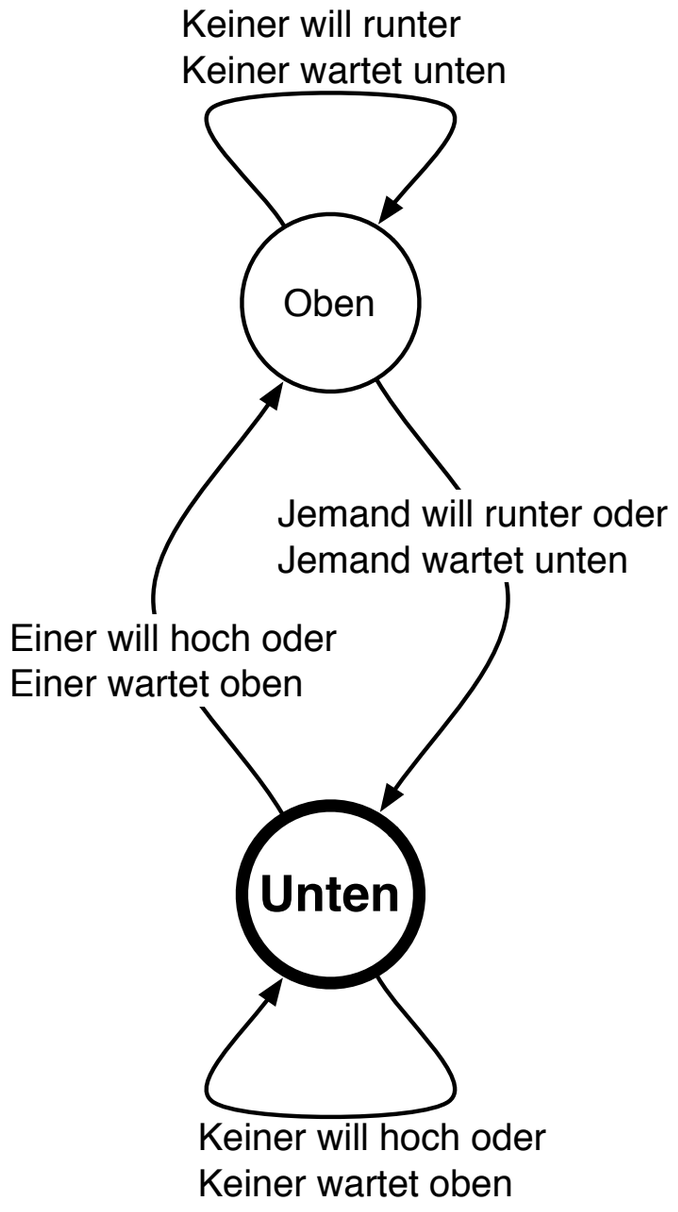




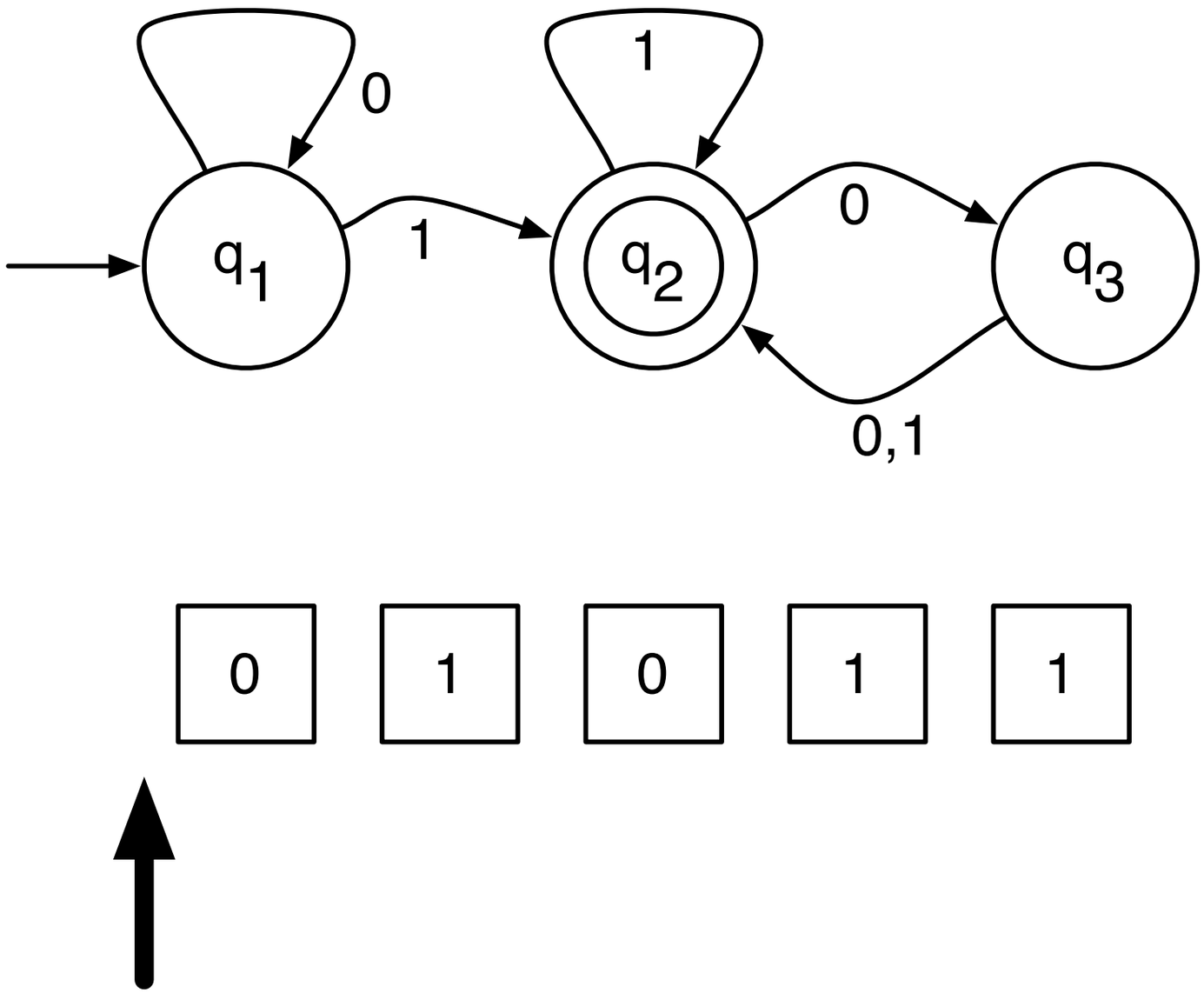
Ein einfacher Automat

- **Lift mit zwei Stockwerken**
  - Eingaben:
    - Im Lift will jemand nach oben
    - Im Lift will jemand nach unten
    - Unten Bedarf
    - Oben Bedarf
  - Zwei Zustände:
    - Lift oben
    - **Lift unten**

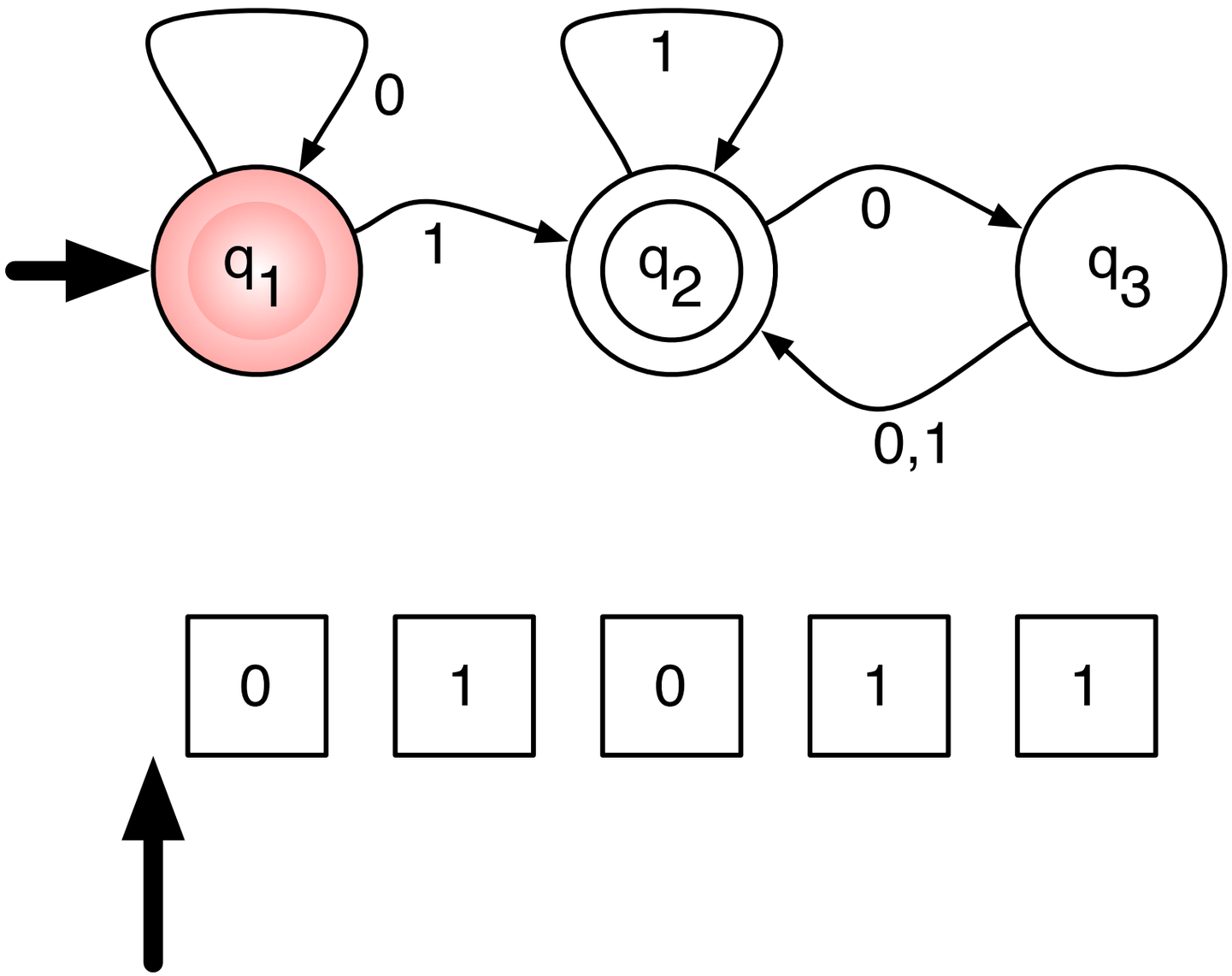




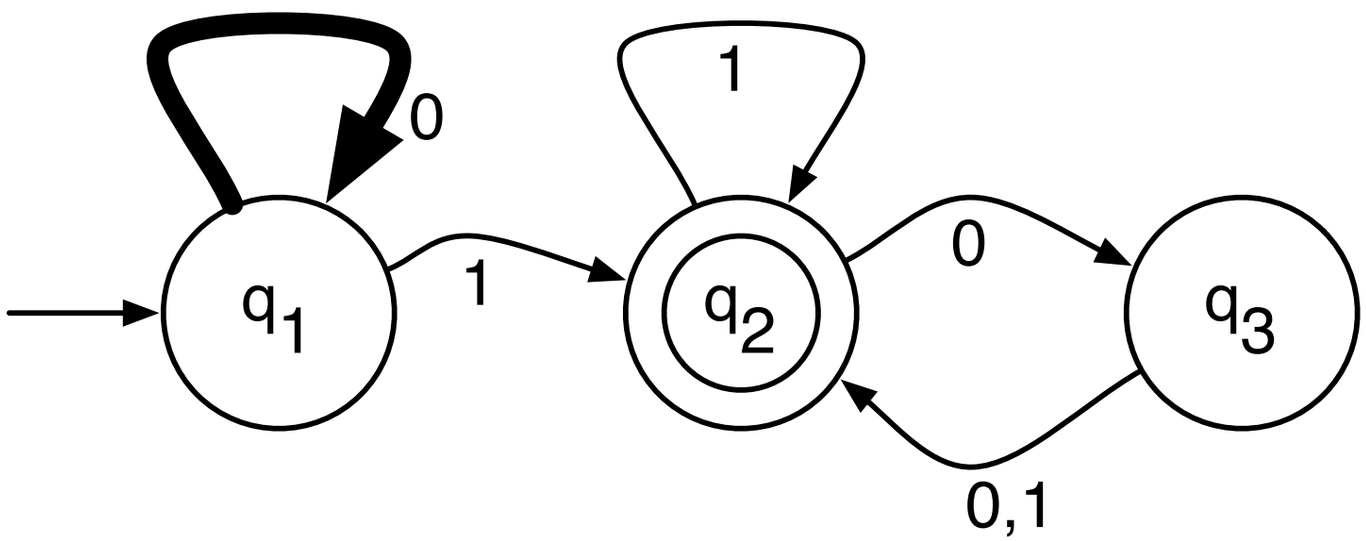
Ein Beispiel

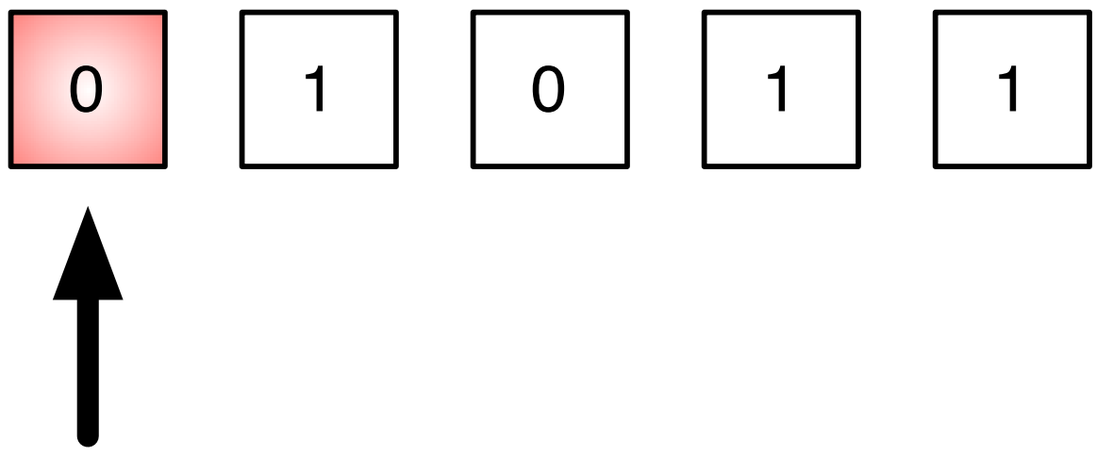


Startzustand q1

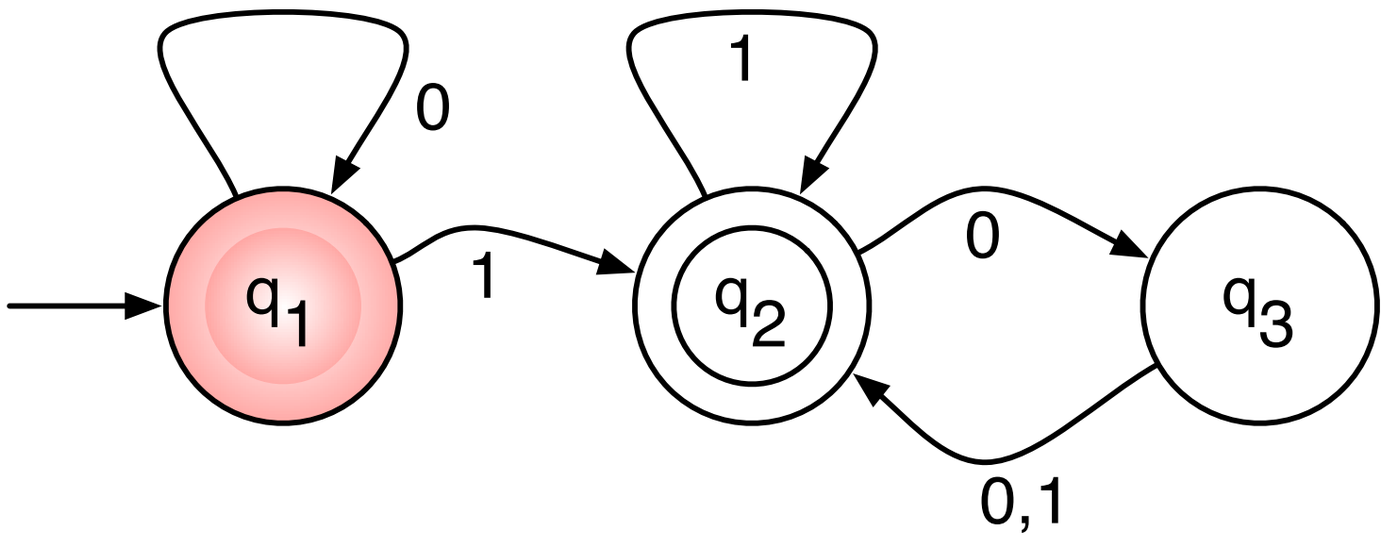


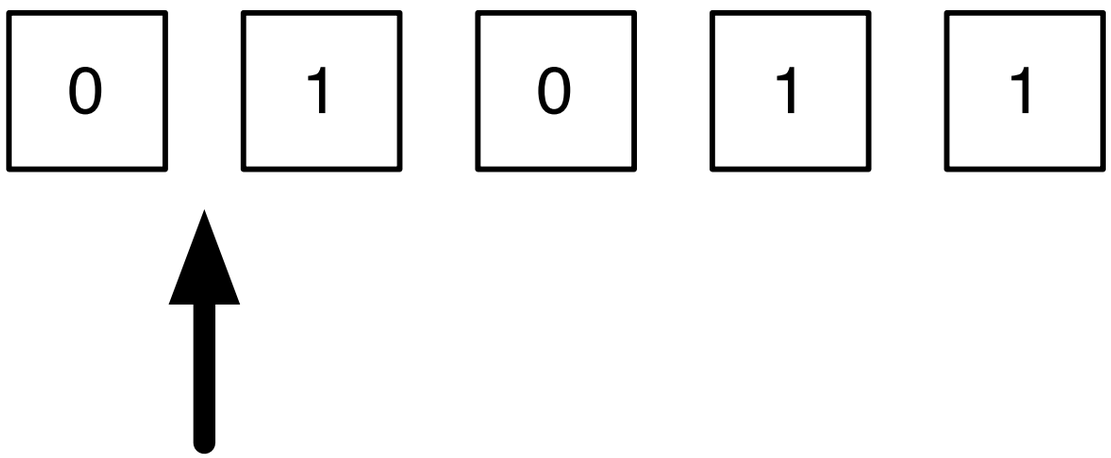
Lesen des ersten Zeichens: 0



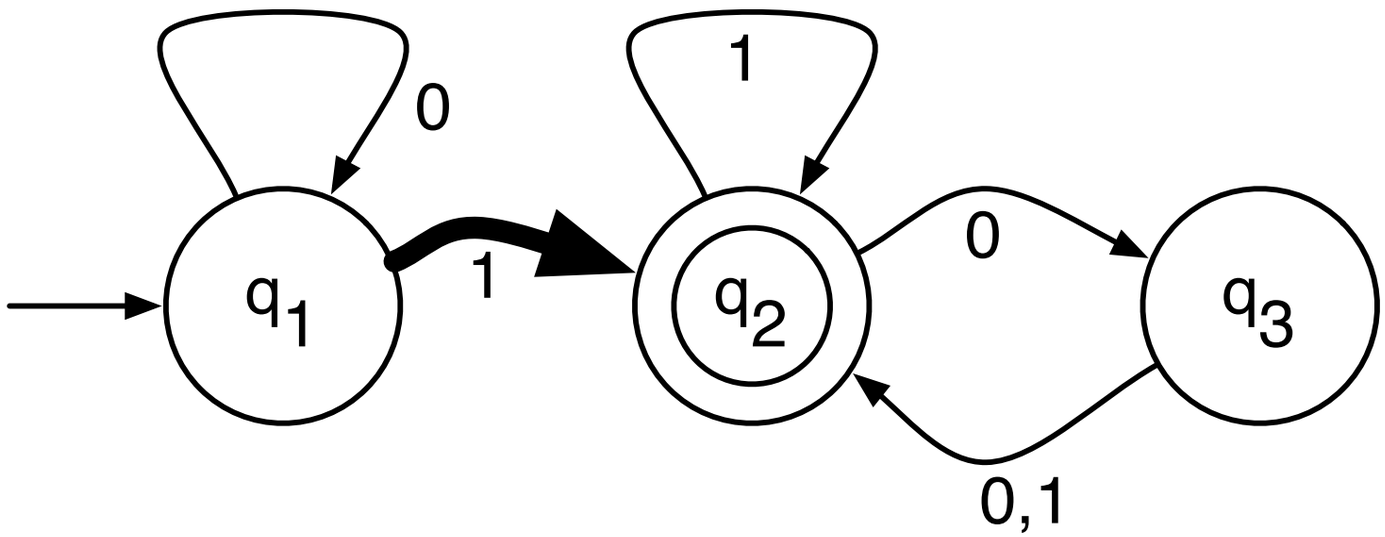


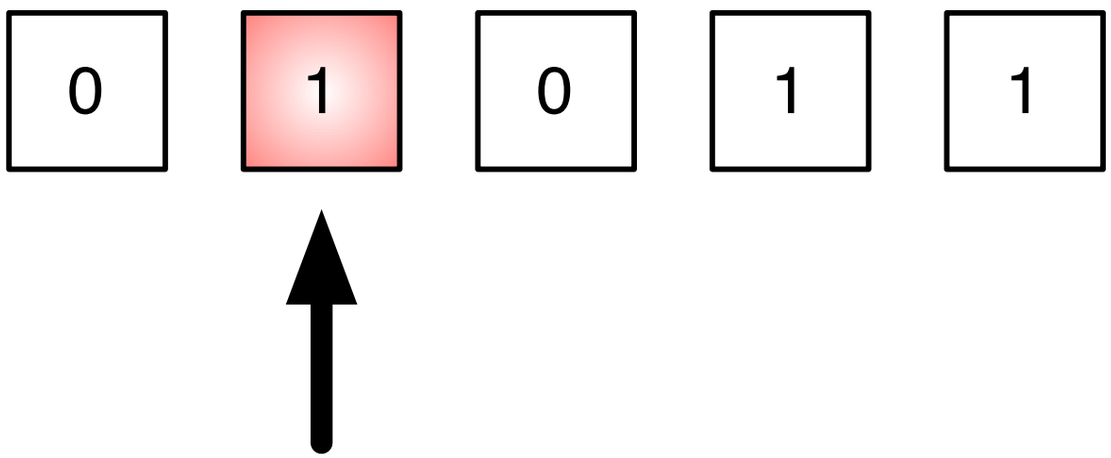
Zustandsübergang zu q1



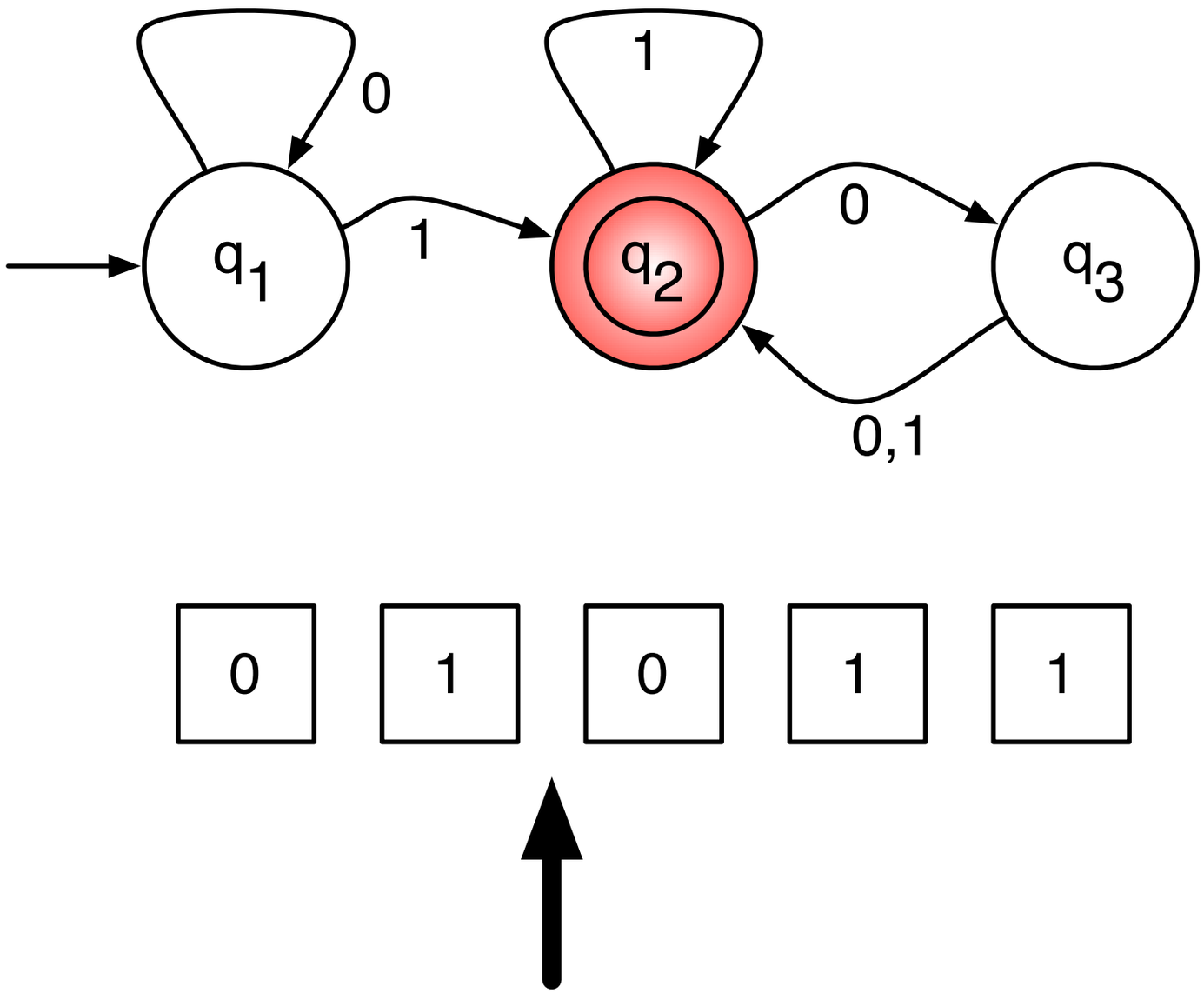


Lesen des 2. Zeichens: 1

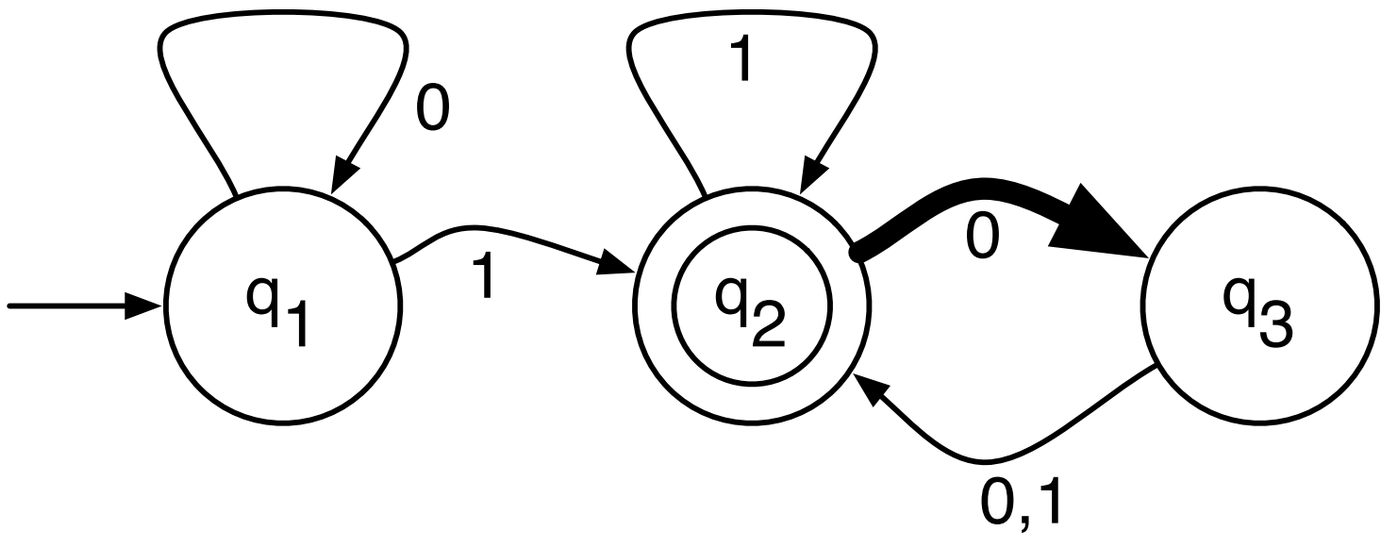


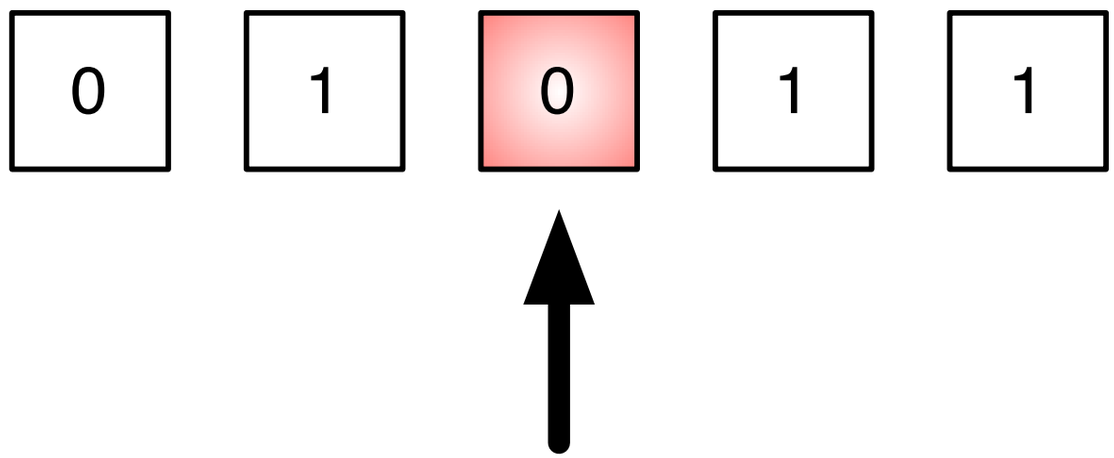


Neuer Zustand: q2

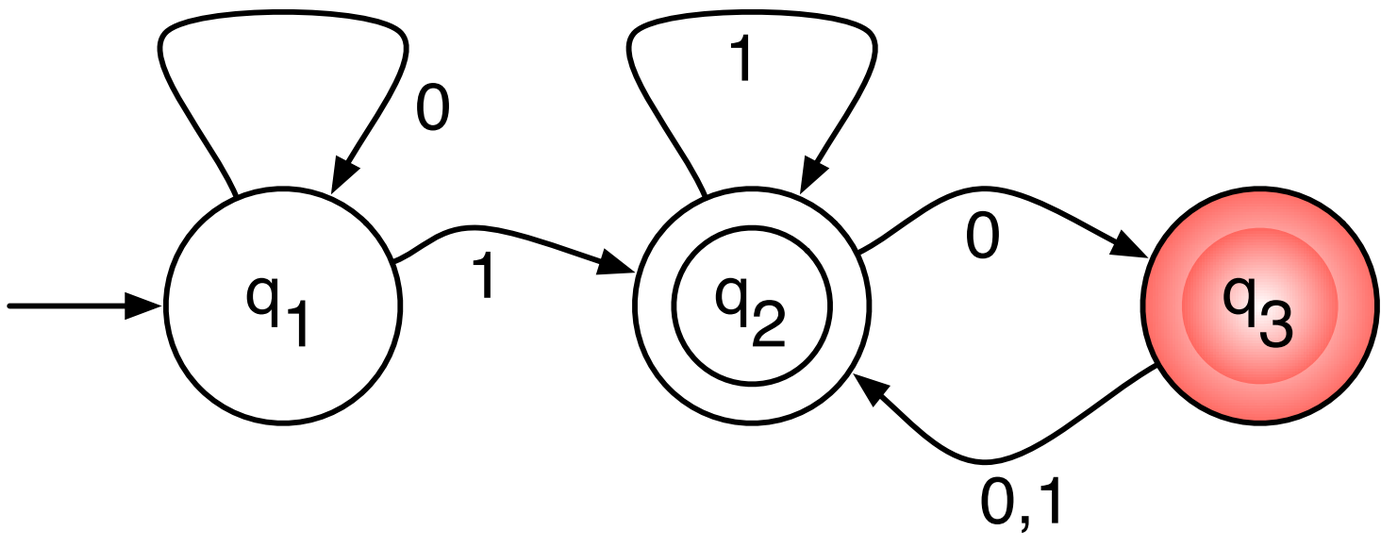


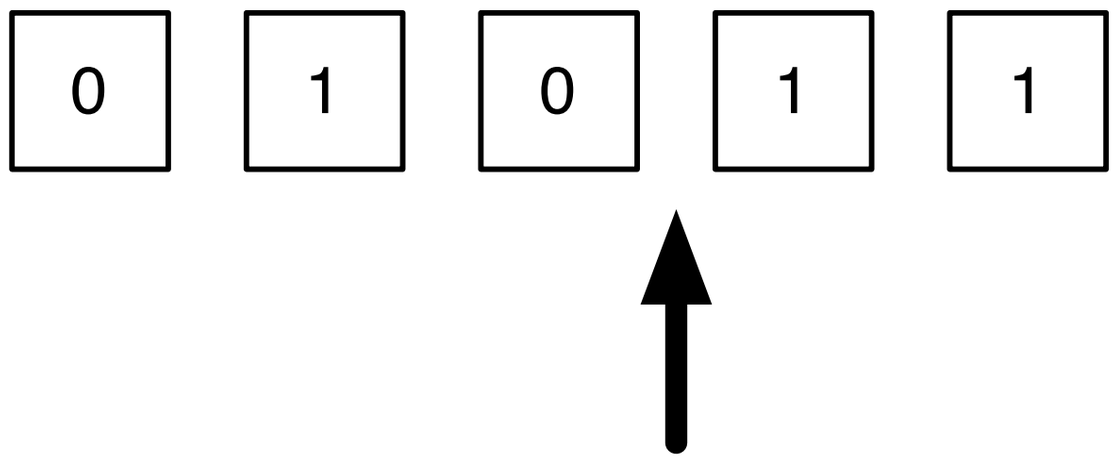
3. Zeichen: 0



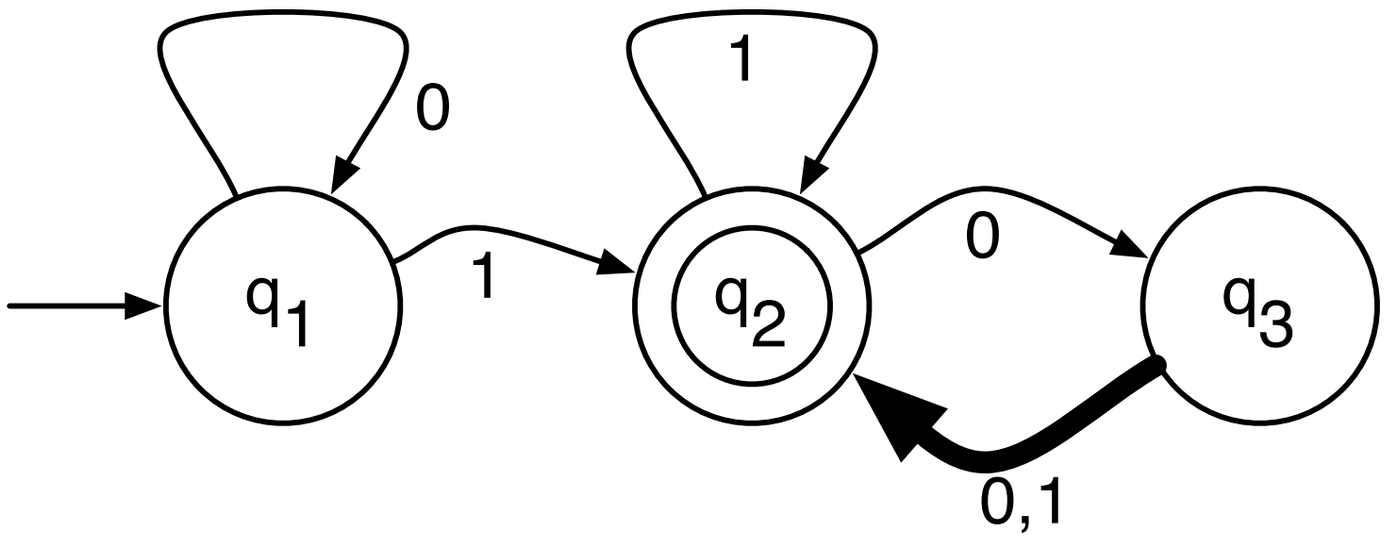


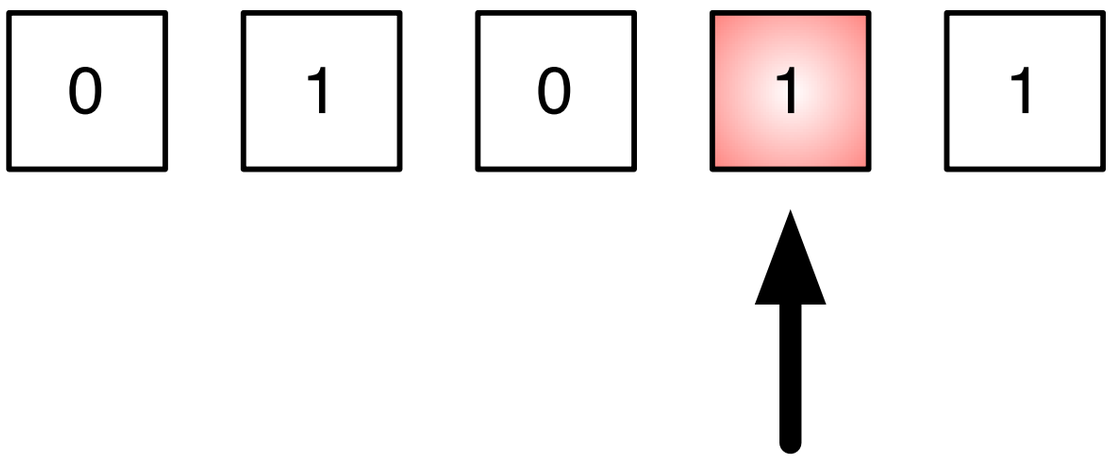
Neuer Zustand: q3



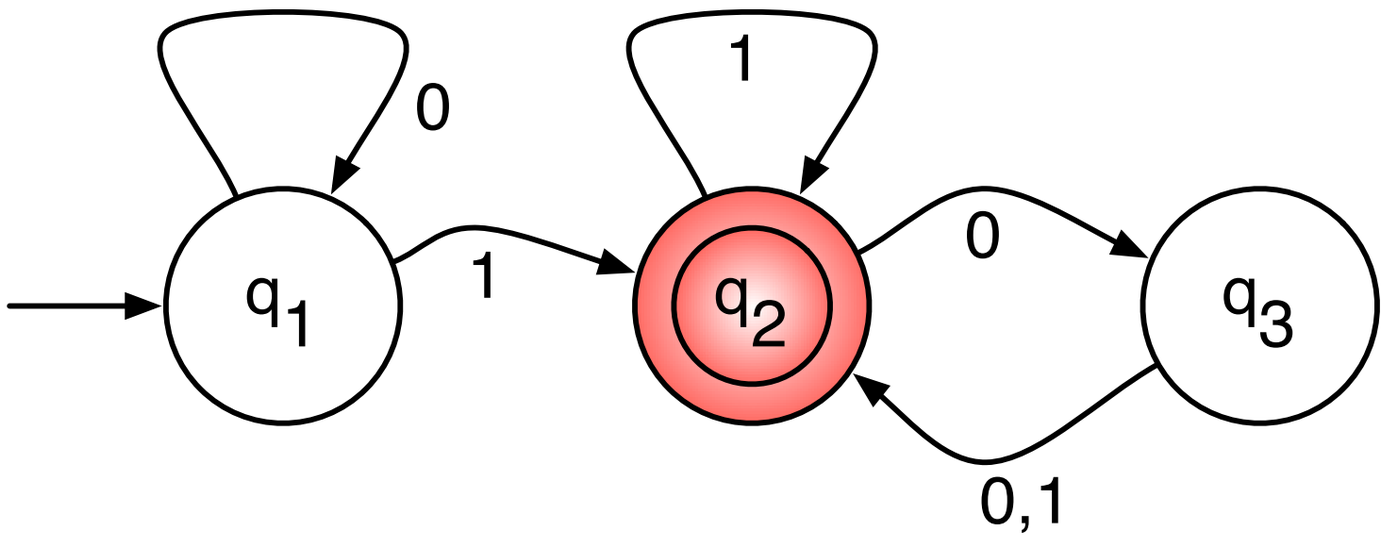


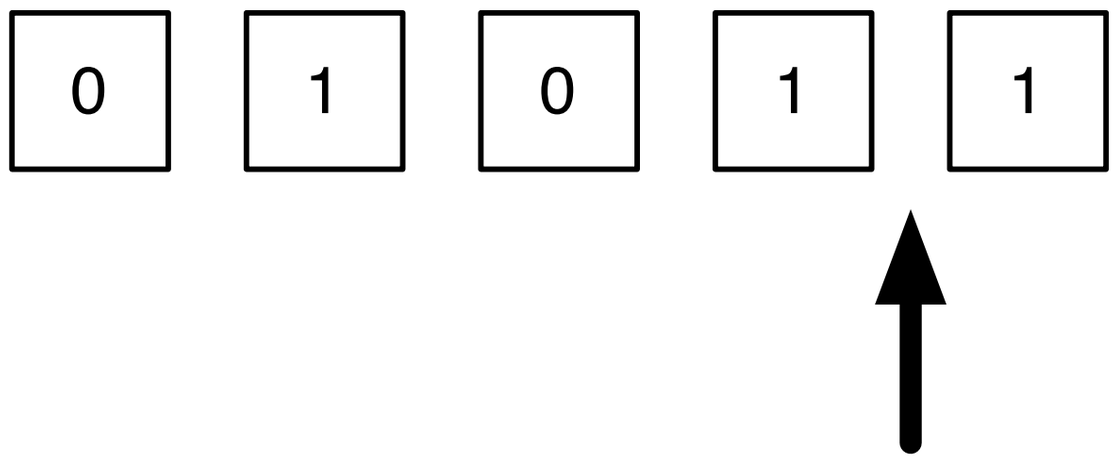
Lesen von “1”



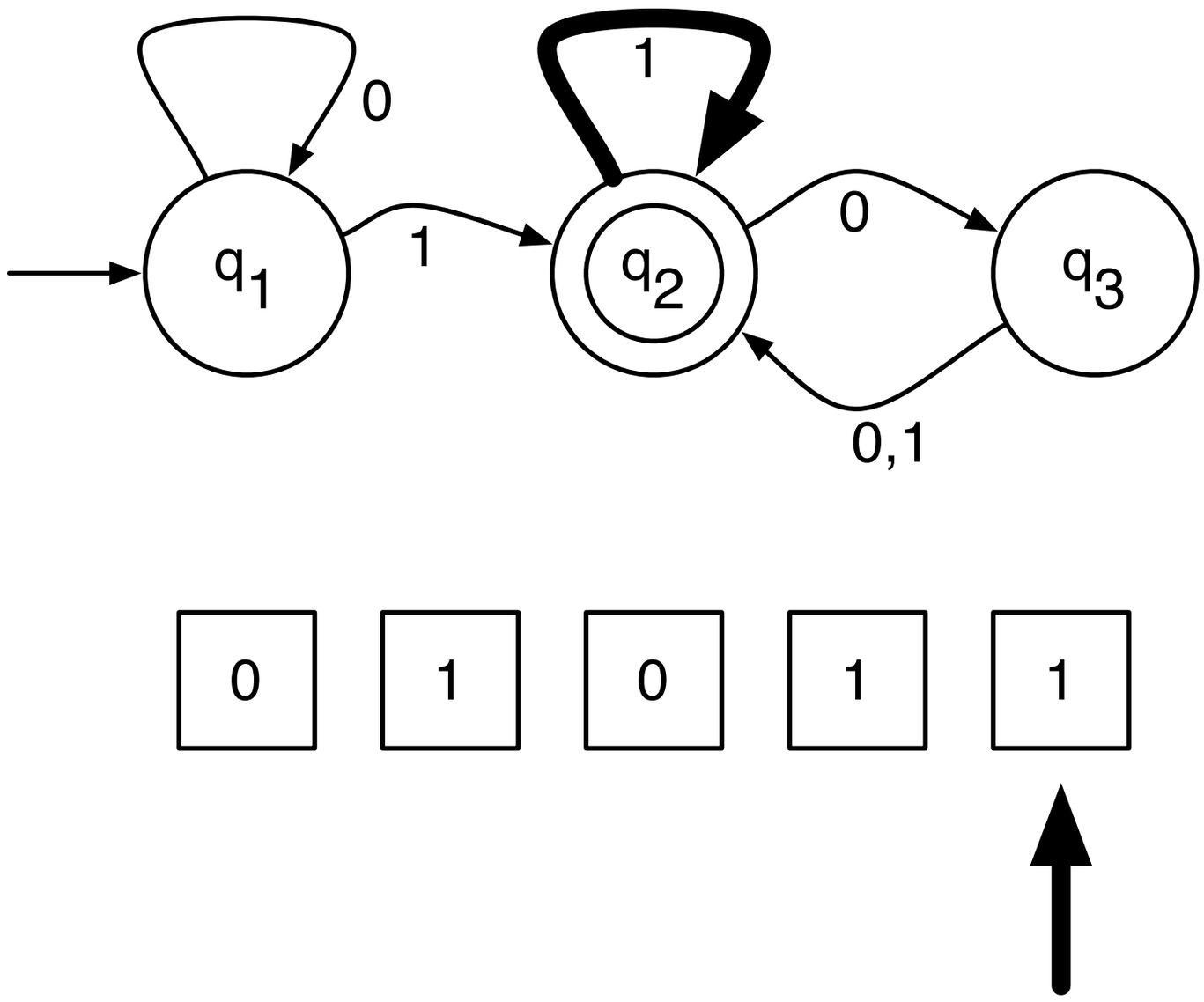


Zustandsübergang zu q2

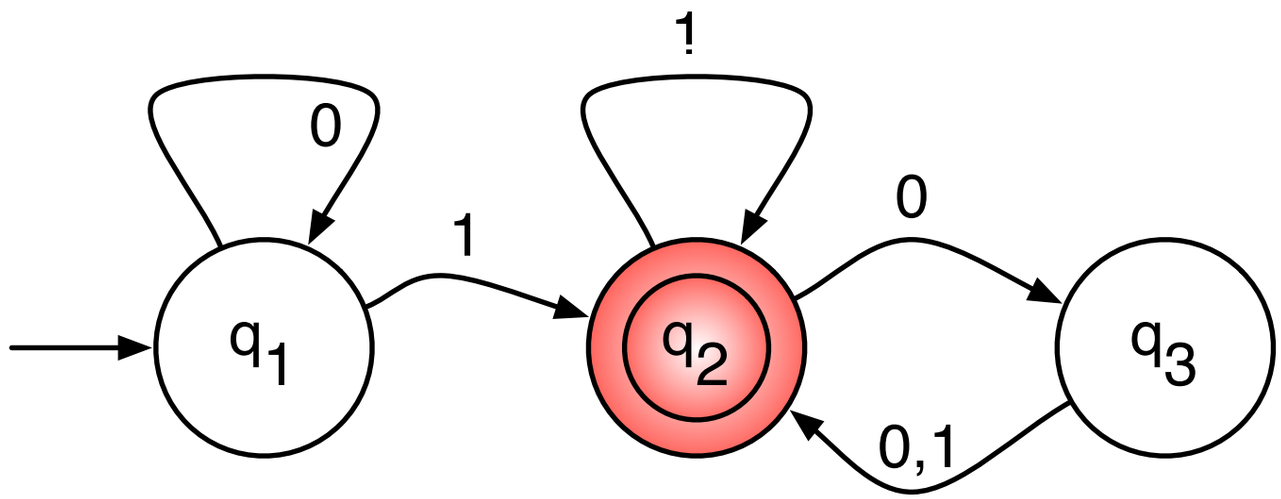


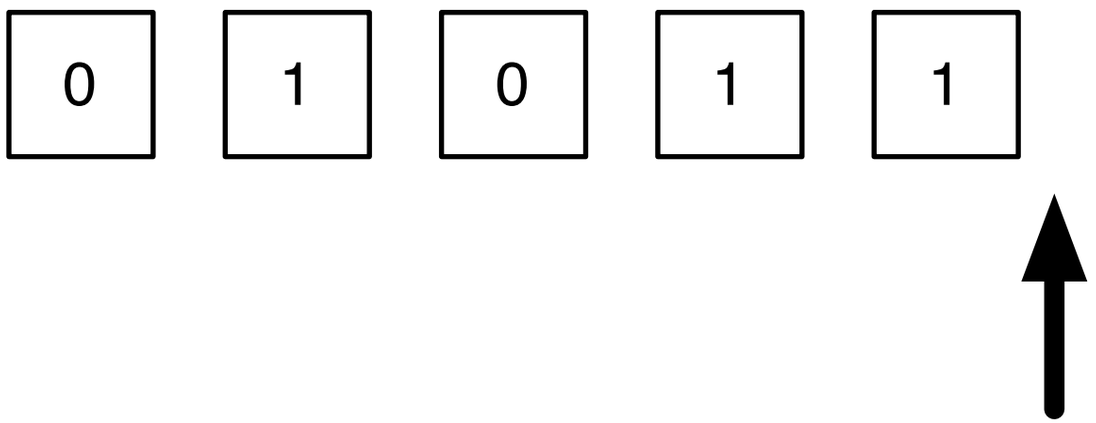


Letztes Zeichen: “1”



Wort “01011” wird akzeptiert





Formale Sprachen und
Endliche Automaten

Worte & Sprachen

Automaten akzeptieren Sprachen

- **Ein Alphabet Σ besteht aus einer endlichen Menge von Zeichen, z.B.**
  - $\Sigma_1 = \{a,b,c\}$
  - $\Sigma_2 = \{0,1\}$
  - $\Gamma = \{0,1,x,y,z\}$
- **Eine Zeichenkette (String/Wort) ist eine endliche Folge (Tupel) von Zeichen, z.B**
  - w = abba
    - Notation: w_1=a, w_2=b, w_3=b, w_4=a
    - Die Länge eines Worts wird mit |w| beschrieben: |w| = 4
- **Σ^* bezeichnet die Menge aller Zeichenketten über Alphabet Σ**
  - z.B.: “abba“ $\in \{a,b\}^*$
  - Die leere Zeichenkette wird mit ε bezeichnet.
    - Es gilt: $|\varepsilon| = 0$
- **Eine Menge von Zeichenketten wird als Sprache bezeichnet**

Worte und Sprachen

- **Worte:**
  - “010010”, “otto”, “abba”
- **Sprachen:**
  - {abba, bab, aa}
  - $\{w \in \{0,1\}^* \mid (\forall i \in \{1,\ldots,|w|\} : w_i = 0) \vee (\forall i \in \{1,\ldots,|w|\} : w_i = 1)\}$
    $= \{\varepsilon, 0, 1, 00, 11, 000, 111, 0000, 1111, ...\}$
  - Palindrom $= \left\{w \in \{a,b\}^* \mid \forall i \in \{1,\ldots,|w|\} : w_i = w_{|w|-i+1}\right\}$
    $= \{\epsilon, a, b, aa, bb, aaa, aba, bab, bbb, \ldots\}$

ε & ∅

- **Sprache ohne Wörter = leere Menge:** ∅
  - ∅ ist kein Wort!
- **Leeres Wort ε ist eine Zeichenfolge der Länge 0**
  - ε ist keine Sprache oder Zeichen
  - Es gibt Sprachen, die ε beinhalten.
  - Es gibt Sprachen, die ε nicht beinhalten.

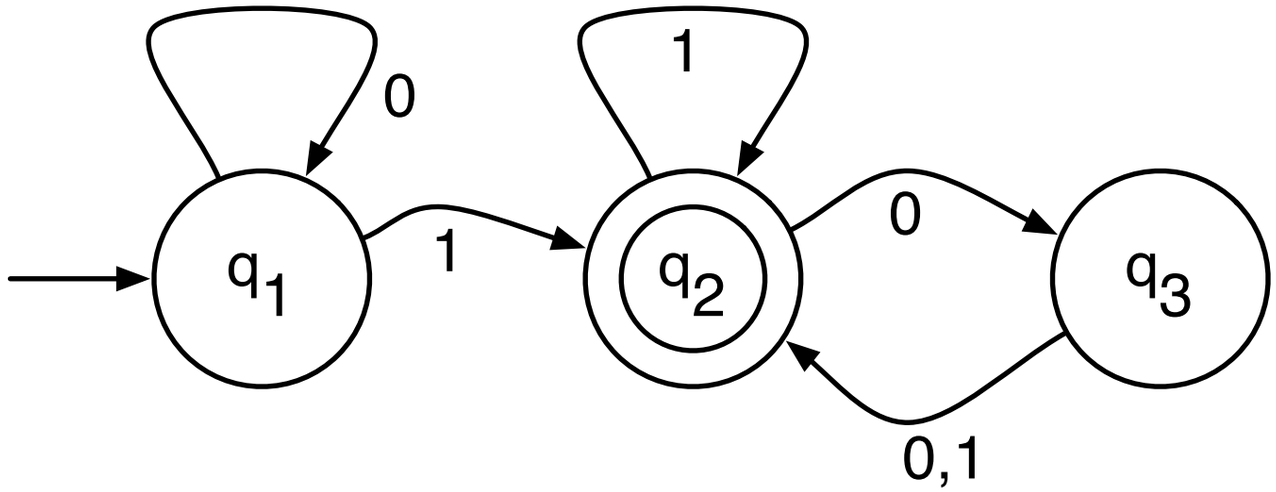
Formale Sprachen und
Endliche Automaten

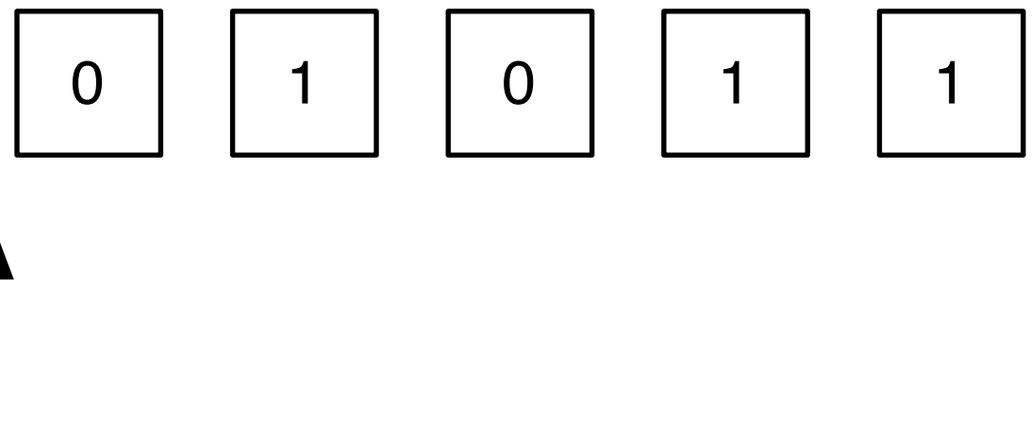
DFA

Komponenten des Beispielautomaten

- **Zustände Q = {q_1, q_2, q_3}**
- **Alphabet Σ = {0,1}**
- **Übergangsfunktion:**
  - $\delta(q_1,0) = q_1$
  - $\delta(q_1,1) = q_2$
  - $\delta(q_2,0) = q_3$
  - $\delta(q_2,1) = q_2$
  - $\delta(q_3,0) = q_2$
  - $\delta(q_3,1) = q_2$
- **Startzustand $q_0 = q_1$**
- **F = {q_2}**

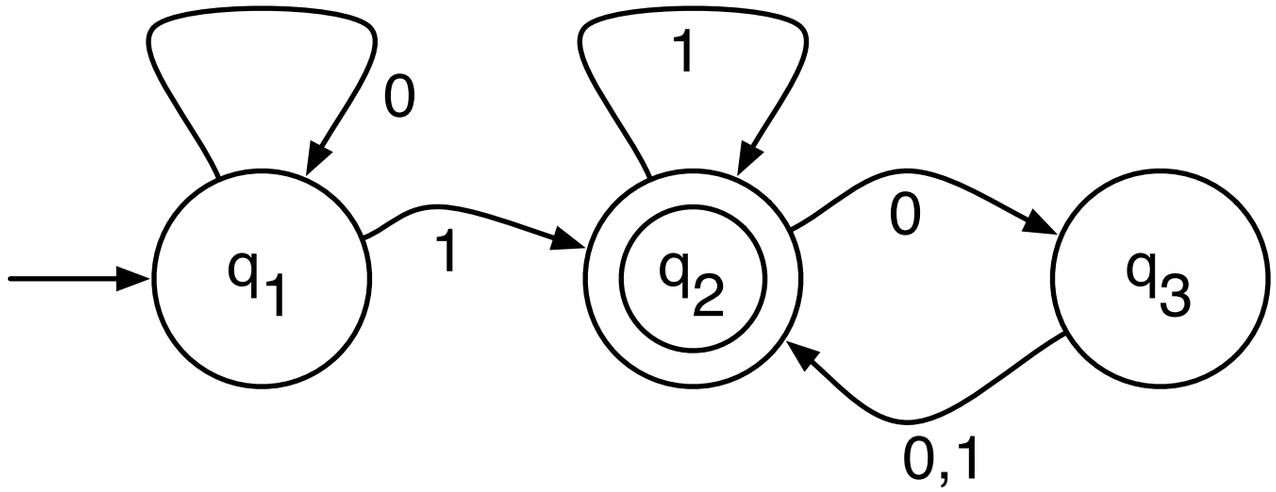
δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

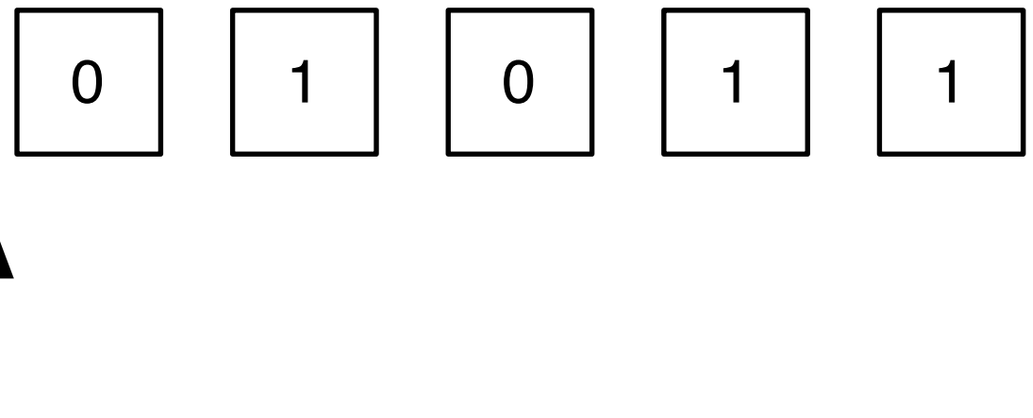




Komponenten des DFA

- **Deterministischer Endlicher Automat= Deterministic Finite Automaton (DFA)**
- **Definition**
  - Ein deterministischer endlicher Automat wird durch die fünf Komponenten (Q,Σ,δ,q_0,F) beschrieben
    - Endliche Zustandsmenge Q
    - Endliches Alphabet Σ
    - Übergangsfunktion $\delta: Q\times\Sigma \rightarrow Q$
    - Startzustand $q_0 \in Q$
    - $F \subseteq Q$: ist die Menge der akzeptierenden Zustände





Die Berechnung eines endlichen Automaten (DFA)

- **Definition**
  - Sei M= (Q, Σ, δ, q_0, F) ein ***endlicher Automat*** und w = $w_1w_2...w_n$ eine Zeichenkette über dem Alphabet Σ.
  - Dann ***akzeptiert M das Wort w*** genau dann, falls es ein Folge r_0r_1 ... r_n von Zuständen aus Q gibt mit
    - $r_0 = q_0$
    - $\delta(r_i,w_{i+1}) = r_{i+1}$, für alle i = 0,...,n-1
    - $r_n \in F$
  - ***M akzeptiert die Sprache A*** falls
    - A = {w | M akzeptiert w }

Formale Sprachen und
Endliche Automaten

Reguläre Sprachen und Operationen

25

Reguläre Sprachen

- **Definition**
  - Eine Sprache heißt regulär, falls ein endlicher Automat sie akzeptiert.
- **Wir schreiben auch REG für die Klasse (Menge) aller regulären Sprachen**

Reguläre Operationen

- **Definition**
  - Die regulären Operationen **Vereinigung**, **Konkatenation** und **Stern** werden wie folgt definiert

1. **Vereinigung**:

$$A \cup B = \{x \mid x \in A \quad \text{oder} \quad x \in B\}$$

2. **Konkatenation**

$$A \circ B = \{x \circ y \mid x \in A \quad \text{und} \quad y \in B\}$$

3. **Stern**

$$A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \quad \text{und} \quad \forall i \in \{1, 2, \ldots, k\} : x_i \in A\}$$

Beispiel

- **Sei Σ = {a,b,c,d,..., z}**
  - A = {wadda, hadda}
  - B = {du, da}
- **Vereinigung**

$$A \cup B = \{wadda, hadda, du, da\}$$

- **Konkatenation**

$$A \circ B = \{waddadu, waddada, haddadu, haddada\}$$

- **Stern**

$$B^* = \{\epsilon, du, da, dudu, duda, dadu, dada, dududu, \ldots\}$$

Formale Sprachen und
Endliche Automaten

Abschluss von REG unter Vereinigung

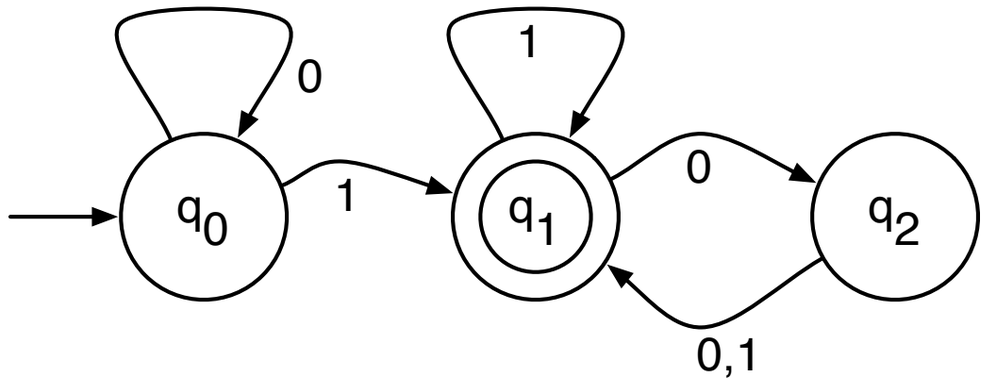
29

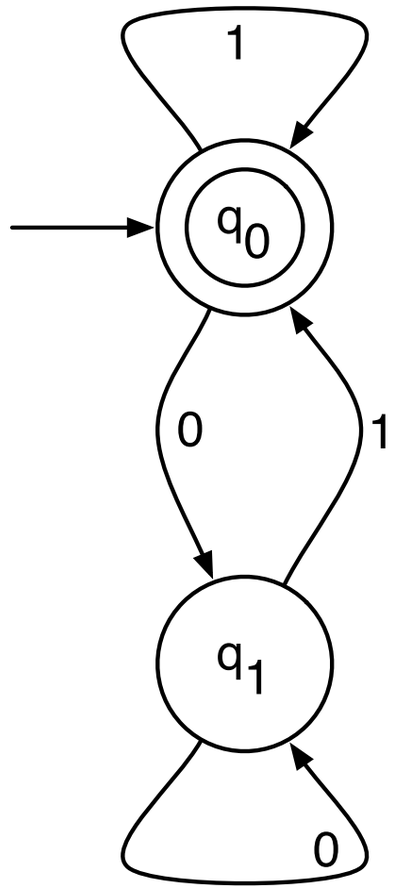
Abschluss unter Vereinigung

- **Theorem**
  - Die Klasse der regulären Sprachen ist abgeschlossen unter der Vereinigung.
- **Falls A und B regulär ist, dann ist auch A ∪ B regulär**
- **D.h.**
  - Gegeben ein endlicher Automat für A und
  - gegeben ein endlicher Automat für B,
    - dann gibt es auch einen für A ∪ B

Beweisidee

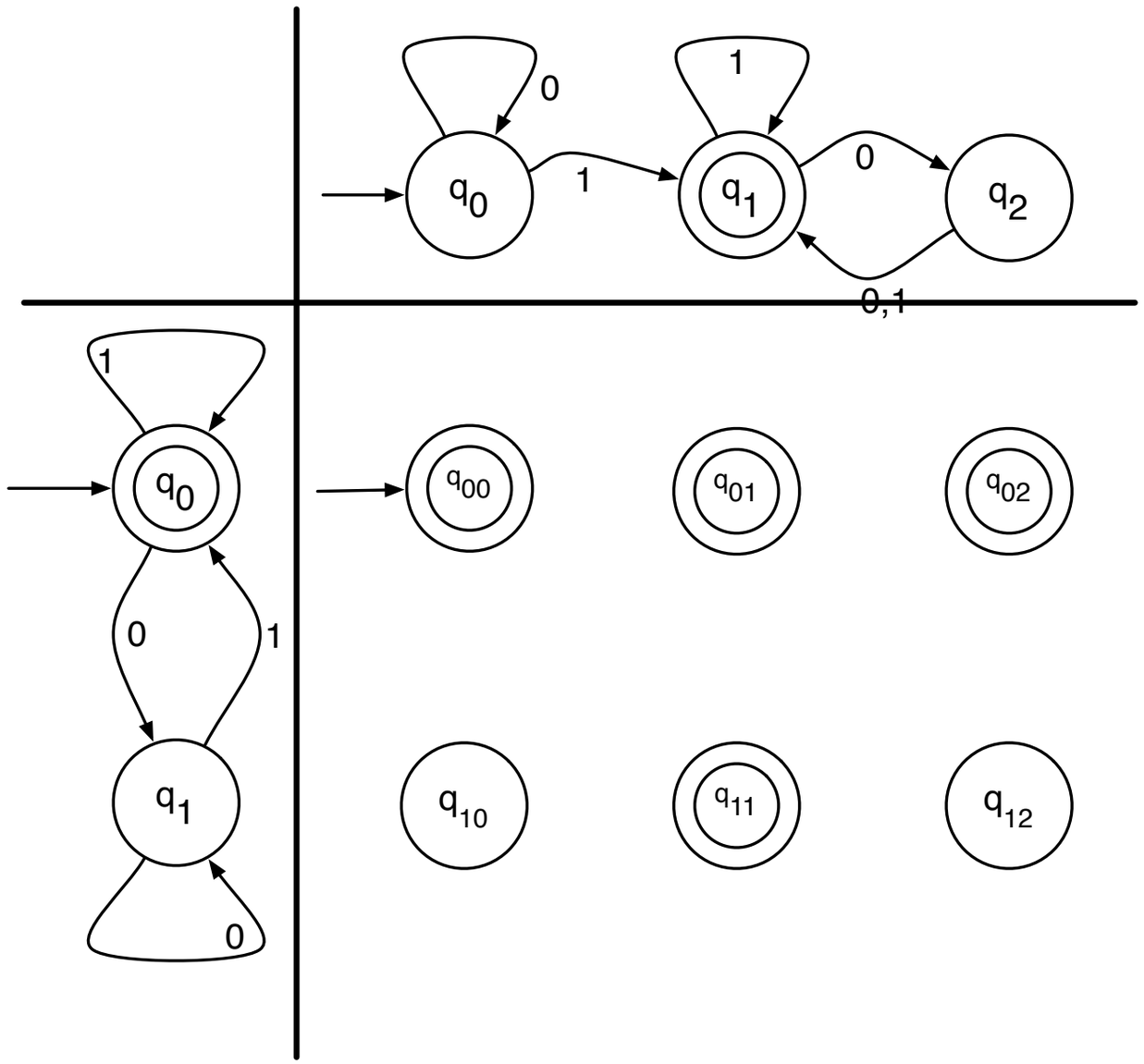
- **Gegeben**
  - 2 Automaten und die Eingabe **0110** Simuliere mit zwei Fingern die Automaten gleichzeitig
- **Falls einer der „Finger“ akzeptiert, akzeptiere**



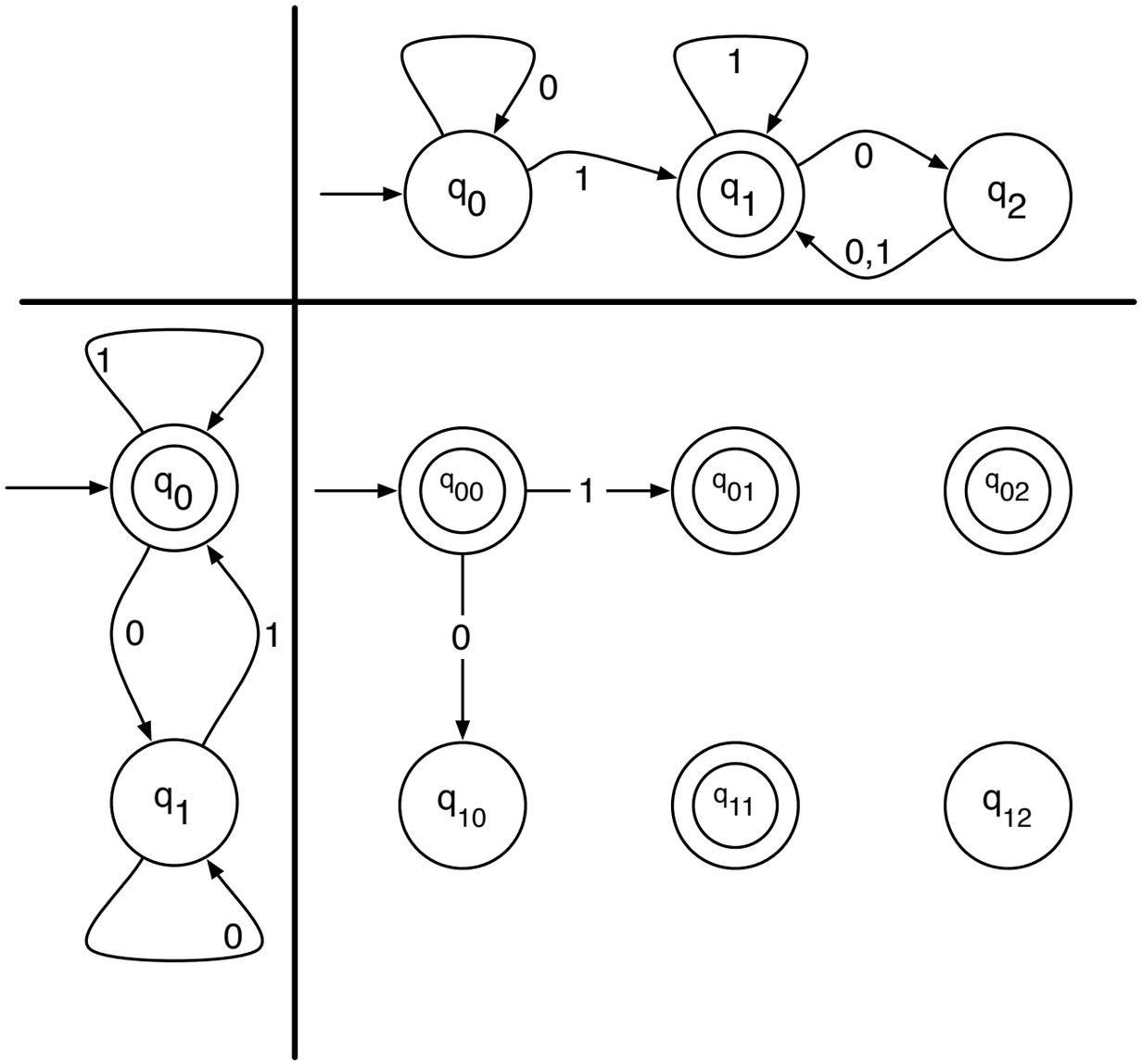


- **Idee:**
  - Idee: Ersetze zwei Finger durch mehr Zustände

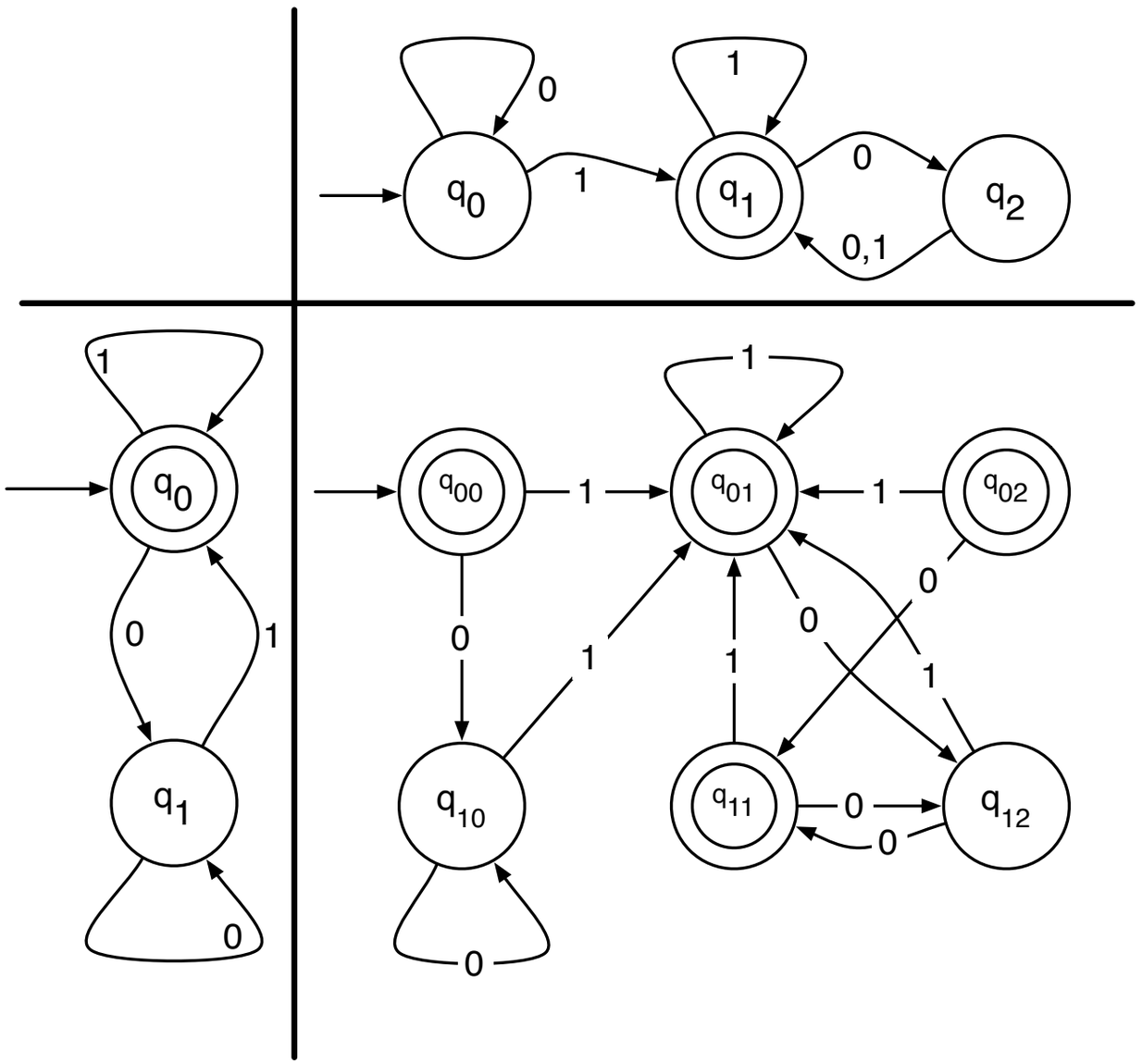
Die Konstruktion des kartesischen Produkts



Ein Zustandsübergang



Automat für die vereinigte Sprache



Die formale Konstruktion

- **Theorem**
  - Gegeben sei ein endlicher Automat $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ für A_1 und
  - gegeben sei ein endlicher Automat $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ für A_2 dann gibt es auch einen endlichen Automaten für $A_1 \cup A_2$
- **Beweis**
  - Wir konstruieren einen Automaten $M = (Q, \Sigma, \delta, q_0, F)$ der $A_1 \cup A_2$ erkennt

1. Zustandsmenge
$Q = \{ (r_1,r_2) \mid r_1 \in Q_1$ und $r_2 \in Q_2\}$,
Q ist das **kartesische Produkt** aus Q_1 und Q_2

2. Alphabet bleibt gleich
Wir erlauben uns diese Vereinfachung, dass die Ausgangsalphabete gleich sind. Eine kleine Modifikation würde den anderen Fall auch beweisen

3. Übergangsfunktion
Für alle $r_1 \in Q_1$ und $r_2 \in Q_2$ und $a \in \Sigma$ gelte

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

4. Anfangszustand: $q_0 = (q_1,q_2)$

5. Akzeptierende Zustände:

$$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ oder } r_2 \in F_2\}$$

Der formale Beweis

- **Theorem**
  - Gegeben sei ein endlicher Automat $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ für A_1 und gegeben sei ein endlicher Automat $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ für A_2 dann gibt es auch einen endlichen Automaten für $A_1 \cup A_2$
- **Beweis**
  - Wir konstruierten einen Automaten $M = (Q, \Sigma, \delta, q_0, F)$ der $A_1 \cup A_2$ erkennt
  - Der Rest des Beweises erfolgt durch eine Induktion über die Länge der Worte
    - **Induktionsanfang**: Maschine ist in q_0 für ε
    - **Induktionsannahme**: Maschine ist in Zustand $(\delta_1(q_1,w), \delta_2(q_2,w))$ nach Abarbeiten von w
      - Erweiterung der Definition der Übergangsfunktion: $\delta_1(q,wa) = \delta_1(\delta_1(q,w),a)$
    - **Induktionsschluss**: Maschine ist in Zustand $(\delta_1(q_1,wa), \delta_2(q_2,wa))$ für w a, wobei $a \in \Sigma$ und $w \in \Sigma^n$
      - Induktionsschritt: Führe eine Transition aus
  - Der Beweis folgt dann aus:

$$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ oder } r_2 \in F_2\}$$

Formale Sprachen und Endliche Automaten

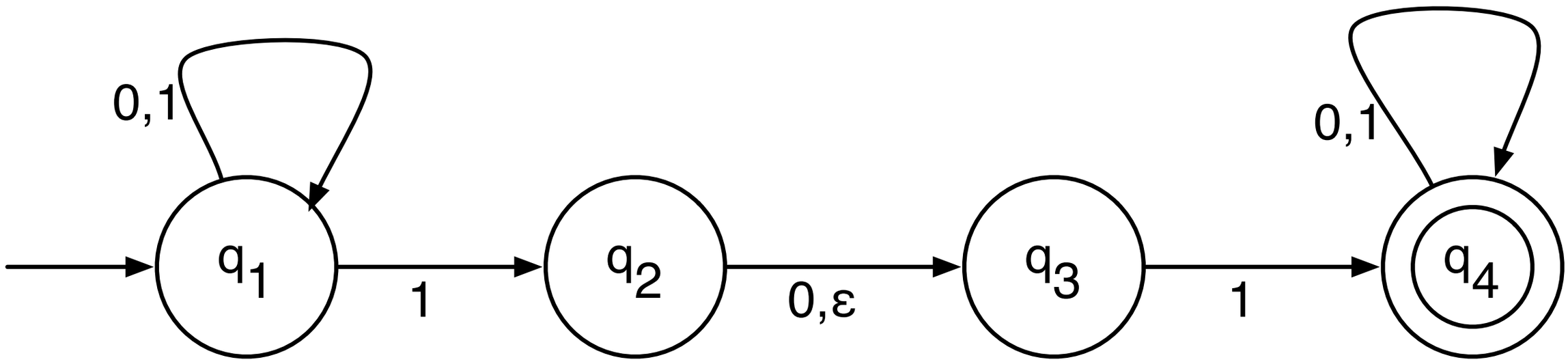
NFA

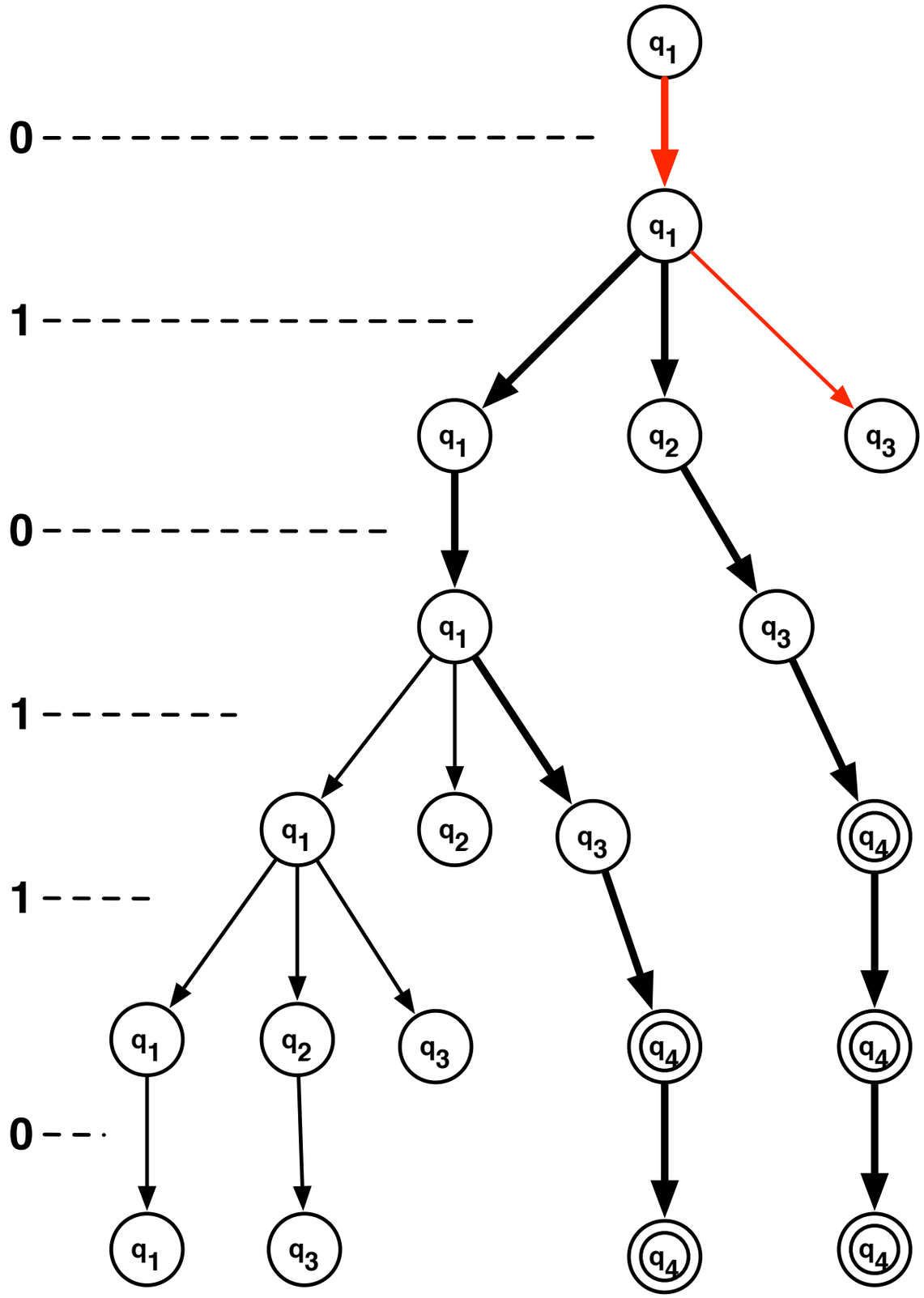
Die Konkatenation

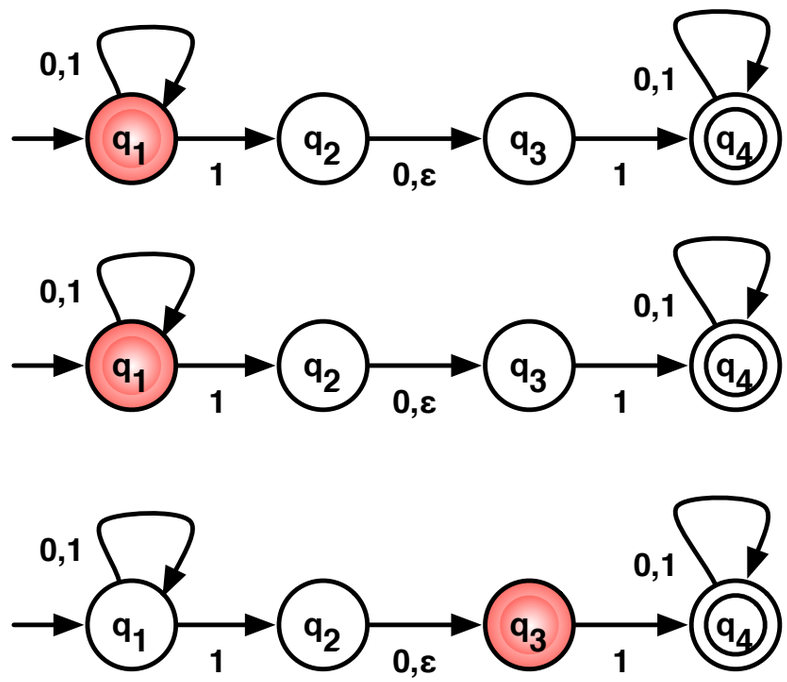
- **Sind endliche Automaten auch unter Konkatenation abgeschlossen?**
- **1. Idee:**
  - Automat M_1 akzeptiert Sprache A_1
  - Automat M_2 akzeptiert Sprache A_2
  - Konstruiere Automat M, der aus den Zuständen von M_1 und M_2 besteht ... ? ....
- **Nach längeren Probieren:**
  - Idee funktioniert nicht mit DFA!!!!
- **Lösung: Betrachte NFA**

Ein nichtdeterministischer Automat

- **Was ist anders?**
  - Von q_1 gehen zwei Übergänge mit 1 aus
  - Von q_2 geht ein ε-Übergang nach q_3
  - Von q_2 fehlt der Übergang bei Zeichen 1
  - Von q_3 fehlt der Übergang bei Zeichen 0
- **Wann akzeptiert so ein Automat?**



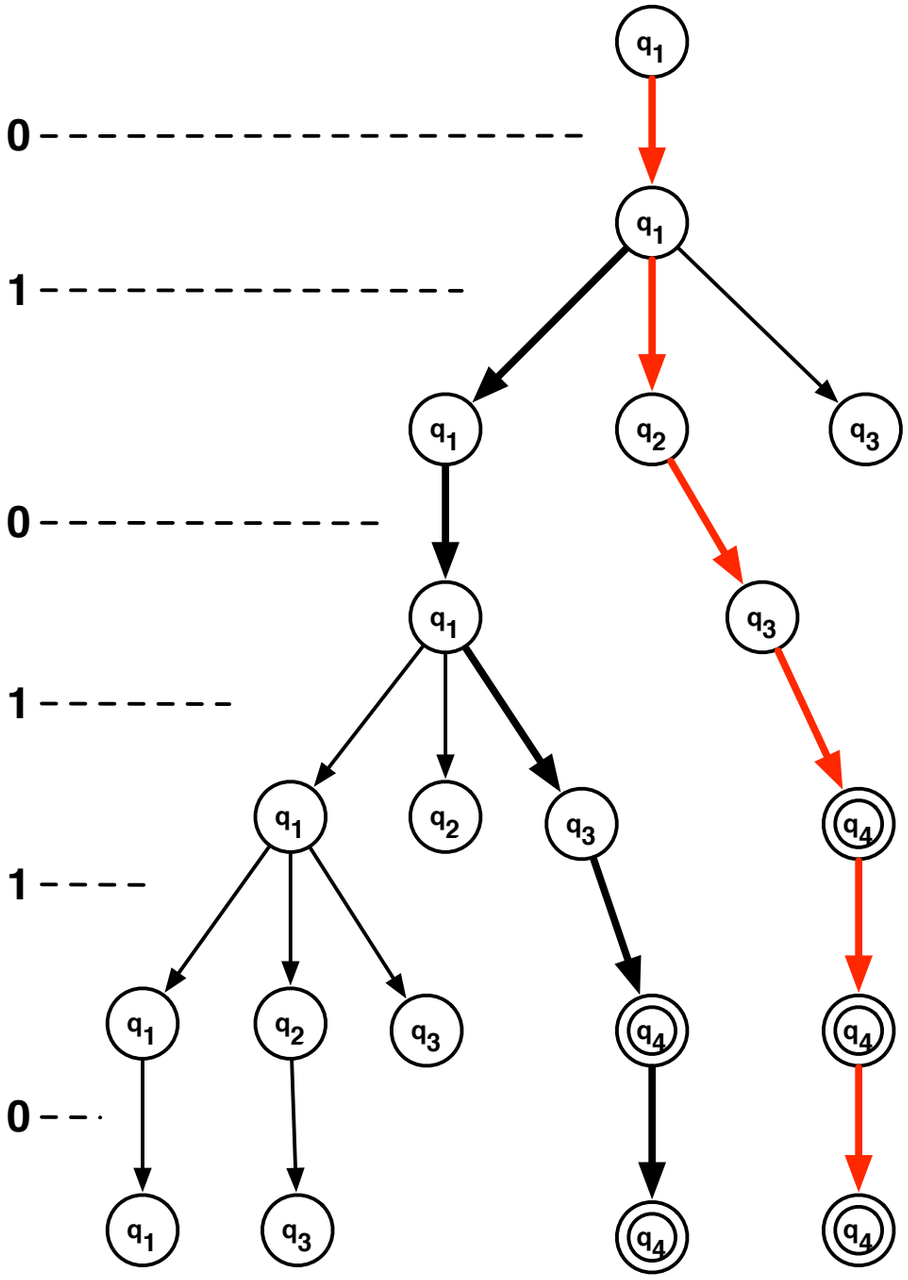


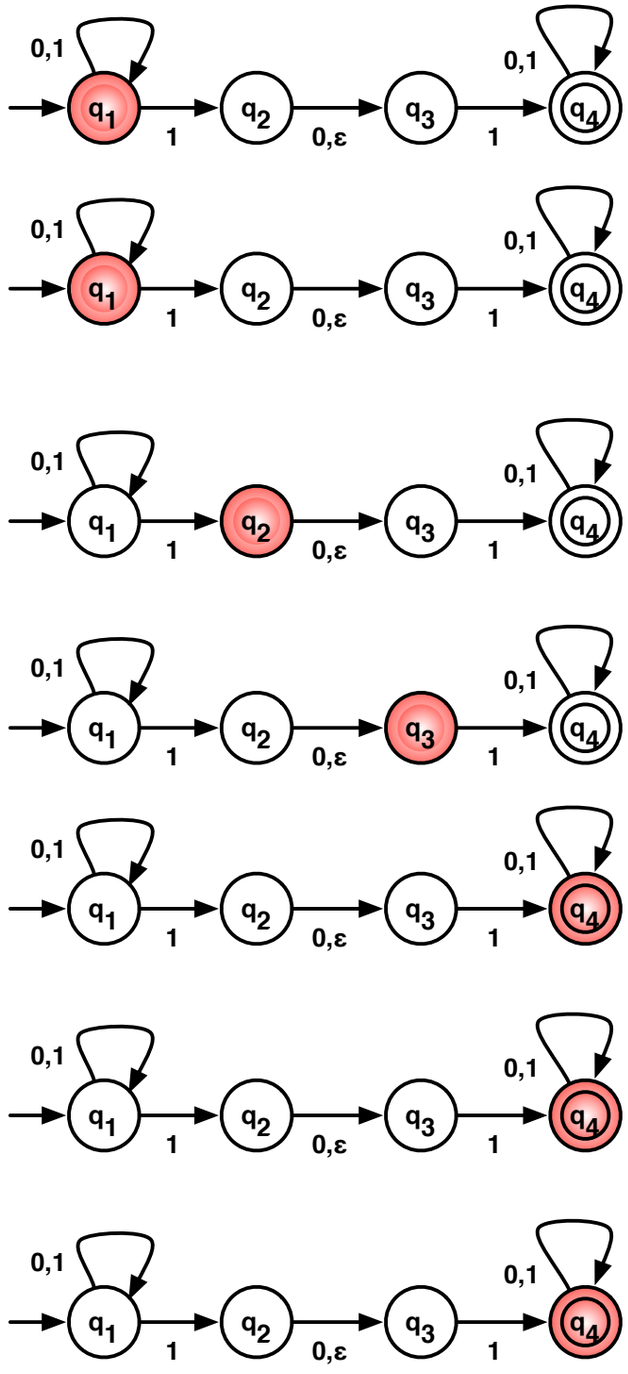


Nicht-akzeptierende Berechnung eines NFA

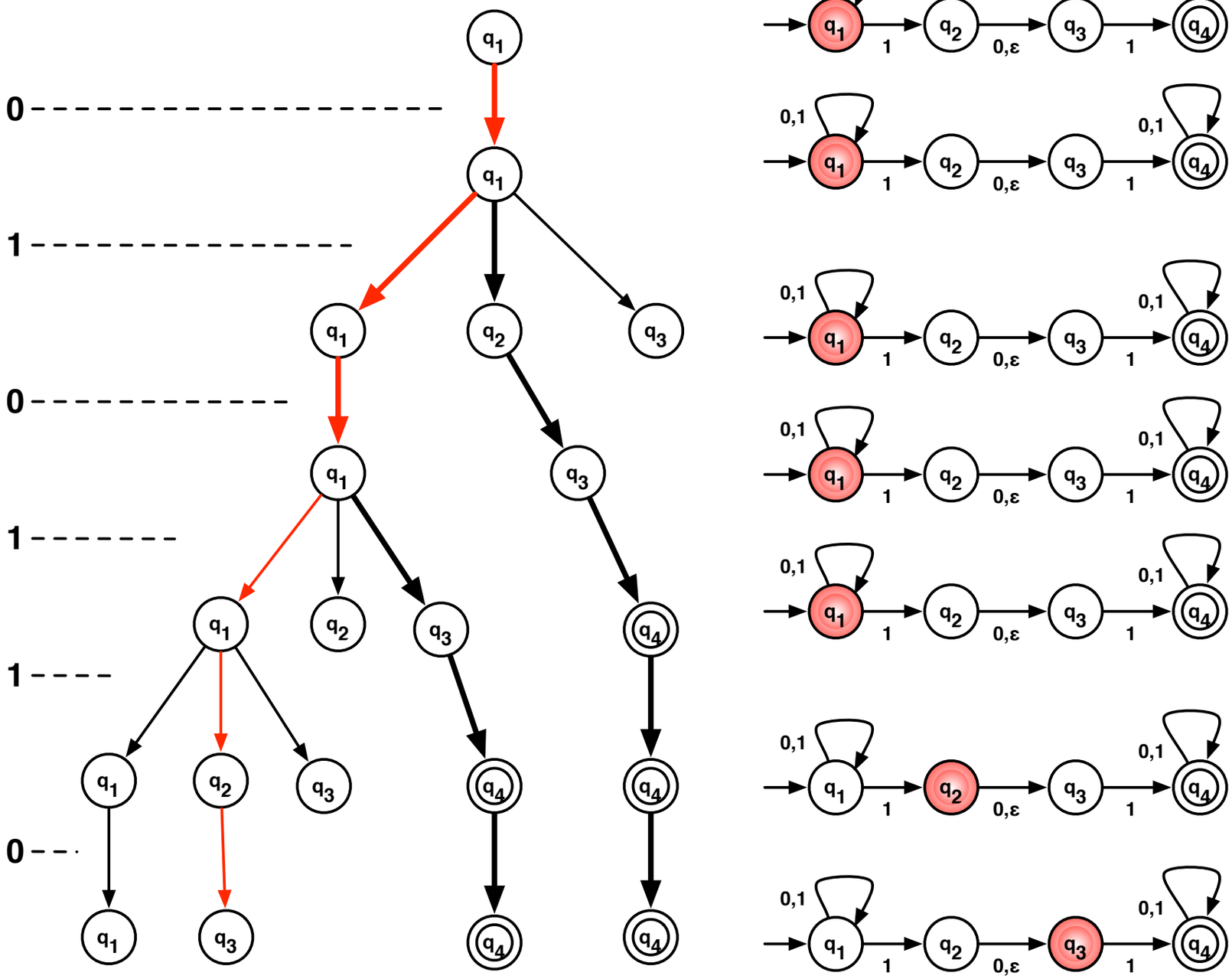
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

Akzeptierende Berechnung eines NFA



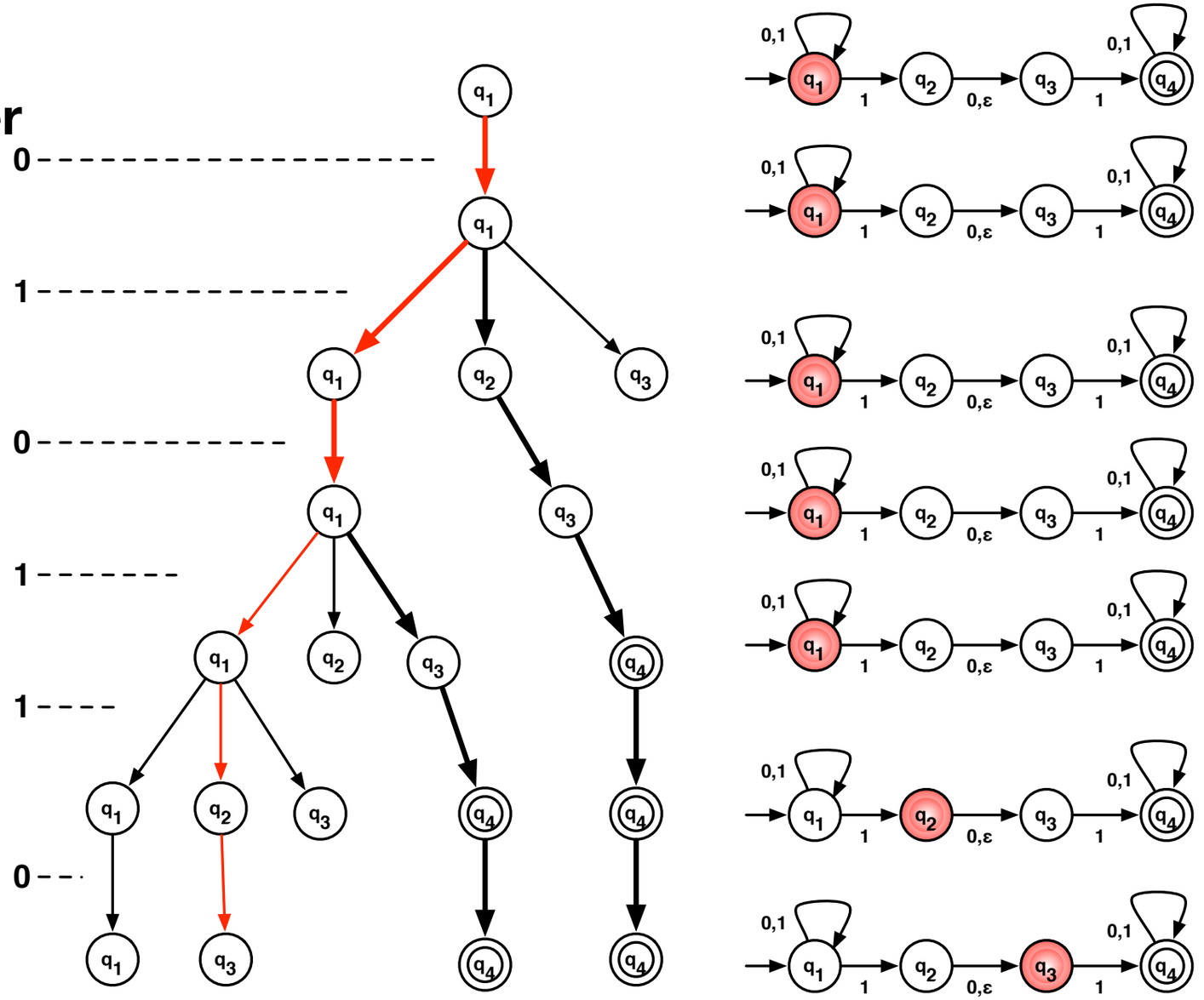


Nicht akzeptierende Berechnung



Wann akzeptiert ein NFA?

- **Ein nicht-deterministischer endlicher Automat (NFA - Nondeterministic Finite Automaton)**
  - akzeptiert ein Wort,
  - falls es (unter allen möglichen) mindestens eine akzeptierende Berechnung gibt.



Komponenten des Beispielautomaten

1. Zustände Q = {q_1, q_2, q_3, q_4}

2. Alphabet Σ = {0,1}

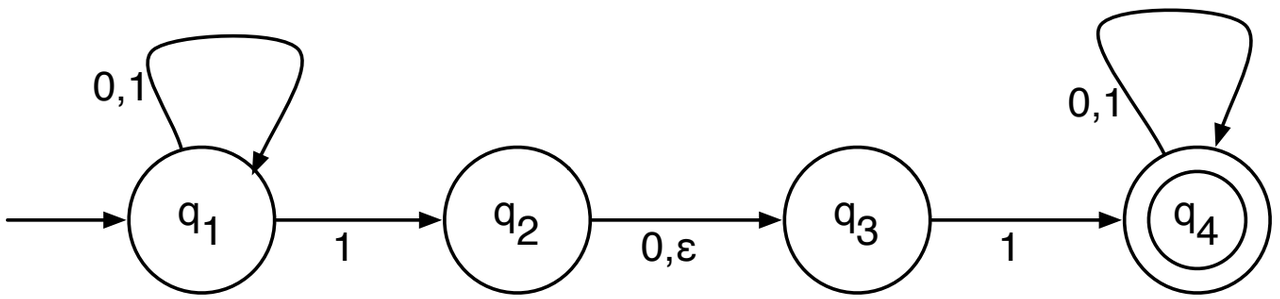
3. Übergangsfunktion:

- $\delta(q_1,0) = \{q_1\}$
- $\delta(q_1,1) = \{q_1, q_2\}$
- $\delta(q_1,\varepsilon) = \varnothing$
- $\delta(q_2,0) = \{q_3\}$
- $\delta(q_2,1) = \varnothing$
- $\delta(q_2,\varepsilon) = \{q_3\}$
- $\delta(q_3,0) = \varnothing$
- $\delta(q_3,1) = \{q_4\}$
- $\delta(q_3,\varepsilon) = \varnothing$
- $\delta(q_4,0) = \{q_4\}$
- $\delta(q_4,1) = \{q_4\}$
- $\delta(q_4,\varepsilon) = \varnothing$

4. Startzustand $q_0 = q_1$

5. Akzeptierender Zustand F = {q_4}

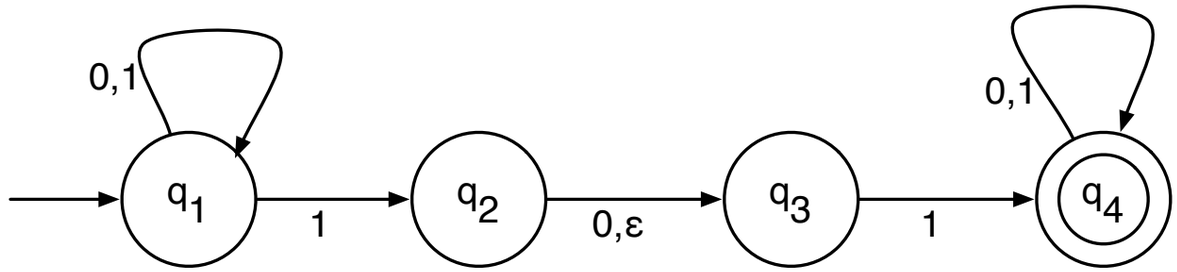
δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
q_2	$\{q_3\}$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$	$\varnothing$



Komponenten eines NFA

- **Definition**
  - Ein nichtdeterministischer endlicher Automat wird durch das 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ beschrieben
    - Q: Eine endliche Menge von Zuständen
    - Σ: ist das Alphabet
    - $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathbf{P}(Q)$ ist die Übergangsfunktion
    - $q_0 \in Q$: ist der Startzustand
    - $F \subseteq Q$: ist die Menge der akzeptierenden Zustände

- **Notation:**
  - Sei $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
  - $\mathbf{P}(M)$ die Potenzmenge von M
    - z.B. $\mathbf{P}(\{a,b\})=\{\varnothing,\{a\},\{b\},\{a,b\}\}$



Berechnung eines NFA

- **Definition**
  - Sei M= (Q, Σ, δ, q_0, F) ein NFA
    Falls es eine
    - Darstellung der Eingabe w = $v_1v_2...v_m$ aus Σ_ε und
    - Folge r_0r_1 ... r_m von Zuständen aus Q gibt, wobei
      - $r_0 = q_0$
      - $r_{i+1} \in \delta(r_i, v_{i+1})$, für alle i ∈ {0, ..., m-1}
      - $r_m \in F$
  - dann akzeptiert M das Wort.

Formale Sprachen und
Endliche Automaten

L(NFA)=L(DFA)

L(DFA) ⊆ L(NFA)

- **Theorem**
  - Jeder deterministische endliche Automat hat einen äquivalenten nichtdeterminstischen Automaten.

- **Beweis**
  - Jeder deterministische endliche Automat **ist** (praktisch) ein nichtdeterministischer endlicher Automat.

L(NFA) ⊆ L(DFA)

- **Theorem**
  - Jeder nichtdeterministischer endliche Automat hat einen äquivalenten determinstischen Automat.

- **Beweis:**
  - Gegeben sei ein nichtdeterministischer endlicher Automat N = (Q, Σ, δ, q_0, F)
  - dann konstruieren wir den DFA M = (Q', Σ, δ', q_0', F') wie folgt:

- **1. Fall: kein ε-Übergang in δ**
  - Zustandsmenge: Q' = P(Q), Q' ist die Potenzmenge von Q
  - Alphabet bleibt gleich
  - Übergangsfunktion
    Für alle R ∈ Q' und a ∈ Σ gelte

$$\delta'(R,a) = \{q \in Q \mid \exists r \in R \ : \ q \in \delta(r,a)\}$$

    andere Notation:

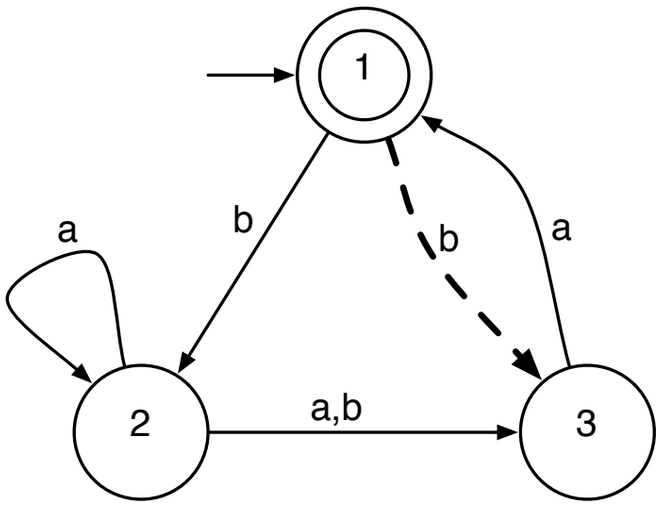
$$\delta'(R,a) = \bigcup_{r \in R} \{\delta(r,a)\}$$

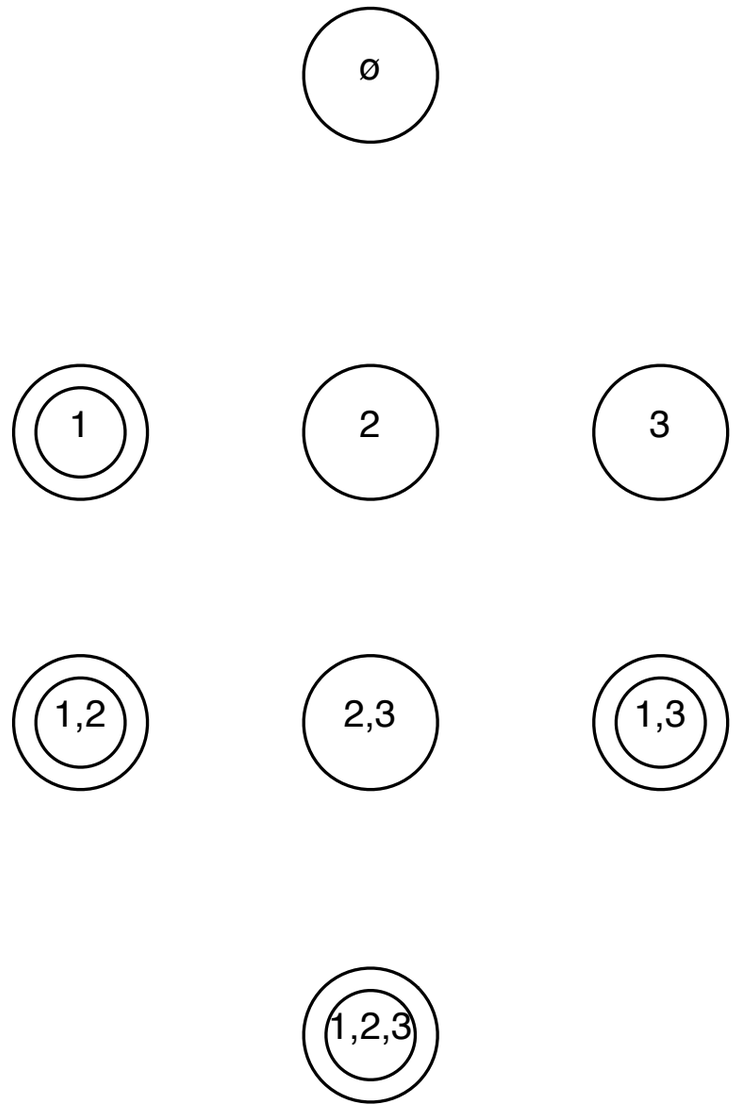
  - Anfangszustand: q_0 = {q_0}
  - Akzeptierende Zustände
    - Zustand R akzeptiert, falls ein akzeptierender Zustand von F in R

$$F' = \{R \in Q' \mid \exists r \in R \ : \ r \in F\}$$

Potenzmenge

- **Zustandsmenge: Q’ = P(Q)**
  - Q’ ist die Potenzmenge von Q





Ohne ε-Übergang

- **Übergangsfunktion**
  - Für alle R ∈ Q‘ und a ∈ Σ gelte

$$\delta'(R,a) = \{q \in Q \mid \exists r \in R \ : \ q \in \delta(r,a)\}$$

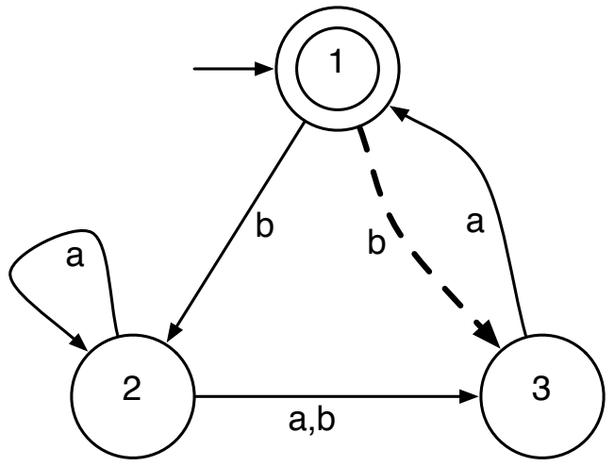
andere Notation:

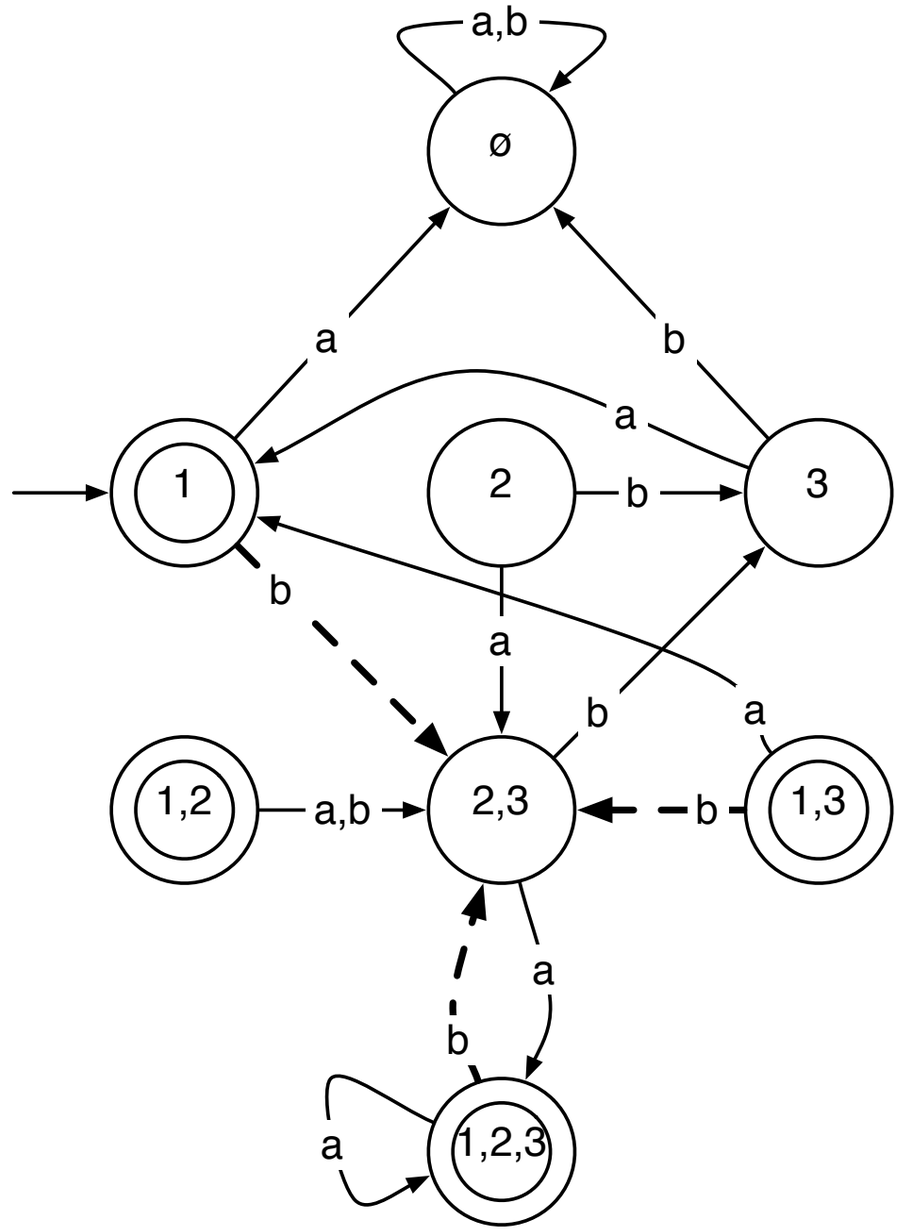
$$\delta'(R,a) = \bigcup_{r \in R} \{\delta(r,a)\}$$

- **Anfangszustand:**
  - $q_0 = \{q_0\}$
- **Akzeptierende Zustände**

$$F' = \{R \in Q' \mid \exists r \in R \ : \ r \in F\}$$

  - Zustand R akzeptiert, falls ein akzeptierender Zustand von F in R ist





L(NFA) $\subseteq$ L(DFA)

- **Gegeben sei ein nichtdeterministischer endlicher Automat N = (Q, Σ, δ, q_0, F)**
  - dann konstruieren wir den DFA M = (Q', Σ, δ', q_0', F') wie folgt:
- **Notation:**

$E(R) = \{q \mid q$ wird von R durch keinen, einen oder mehrere ϵ-Übergänge erreicht$\}$

- **2. Fall: Mit ε-Übergang in δ**
  - Zustandsmenge: Q' = **P**(Q), Q' ist die Potenzmenge von Q
  - Übergangsfunktion
    - Für alle R ∈ Q' und a ∈ Σ gelte

$$\delta'(R,a) = \{q \in Q \mid \exists r \in R \ : \ q \in E(\{\delta(r,a)\})\}$$

andere Notation:

$$\delta'(R,a) = \bigcup_{r \in R} E(\{\delta(r,a)\})$$

- Anfangszustand: q_0 = E({q_0})
- Akzeptierende Zustände:
  - Zustand R akzeptiert, falls ein akzeptierender Zustand von F in R ist

$$F' = \{R \in Q' \mid \exists r \in R \ : \ r \in F\}$$

Mit ε-Übergang

- **Übergangsfunktion**
  - Für alle R ∈ Q‘ und a ∈ Σ gelte

$$\delta'(R,a) = \{q \in Q \mid \exists r \in R \ : \ q \in E(\{\delta(r,a)\})\}$$

andere Notation:

$$\delta'(R,a) = \bigcup_{r \in R} E(\{\delta(r,a)\})$$

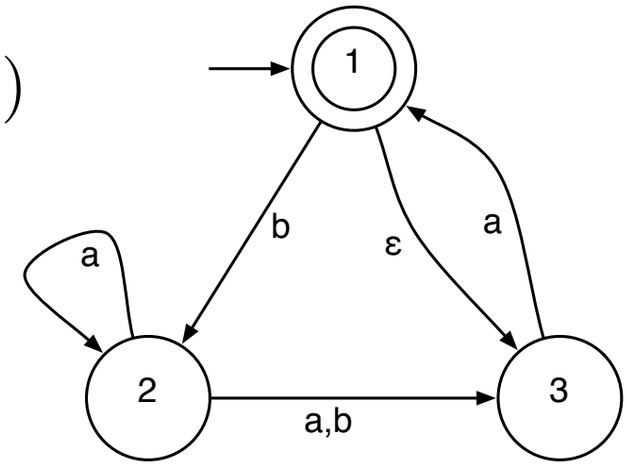
- **Anfangszustand:**

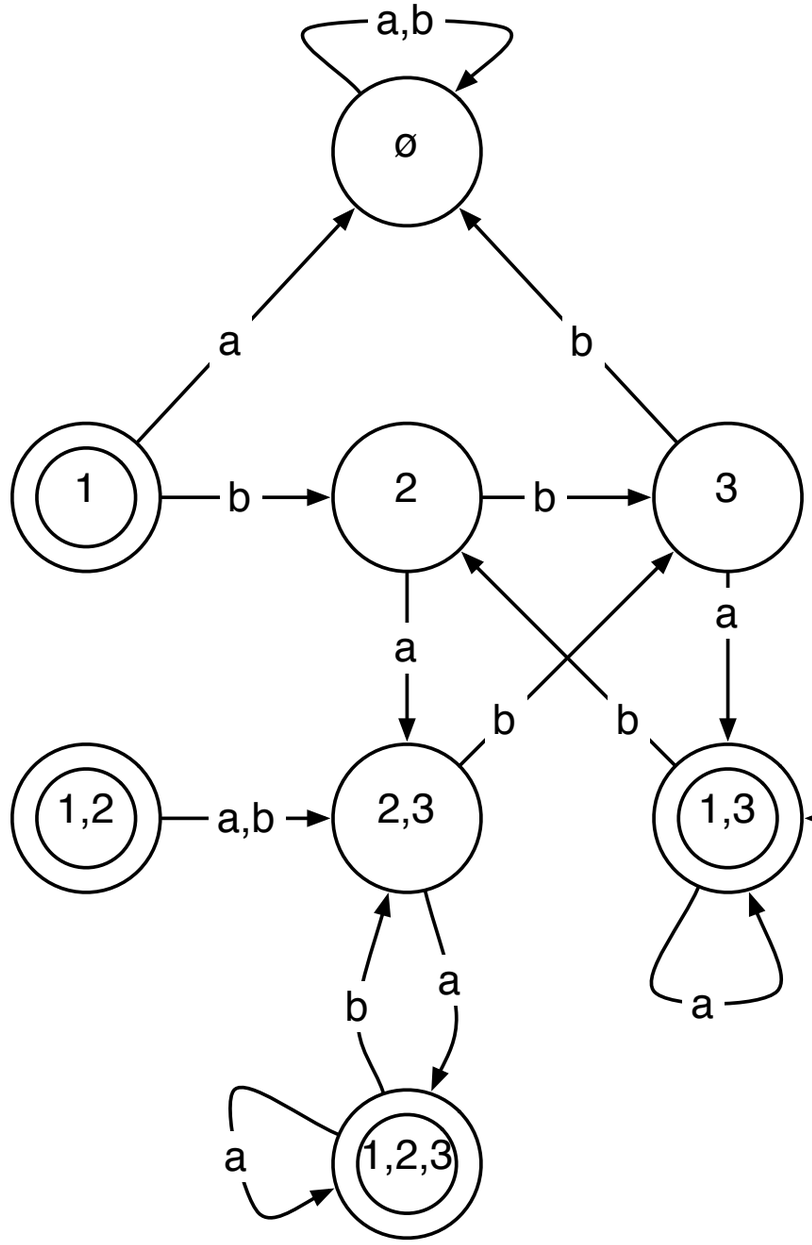
$$q_0' = E(\{q_0\})$$

- **Akzeptierende Zustände**

$$F' = \{R \in Q' \mid \exists r \in R \ : \ r \in F\}$$

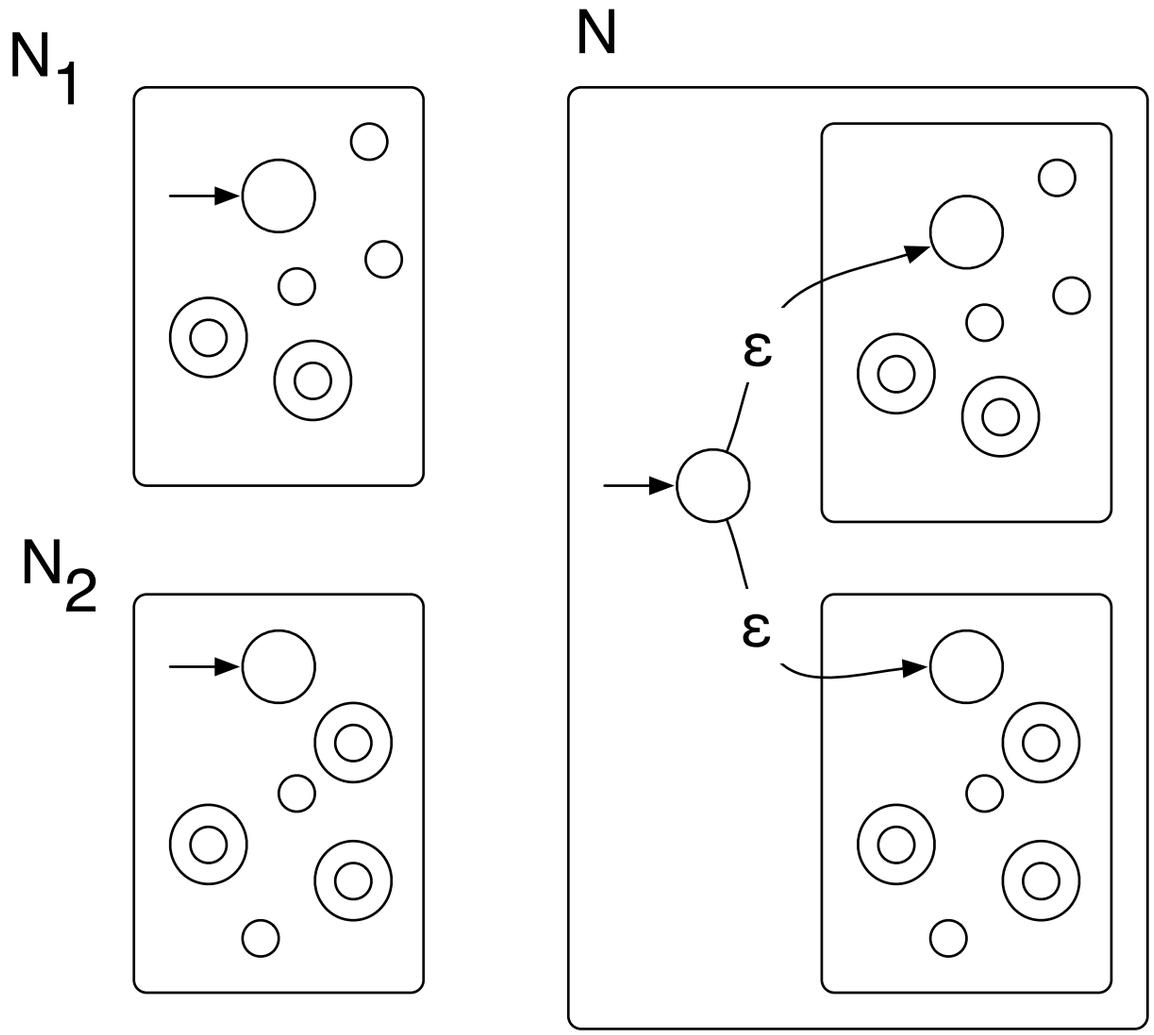
  - Zustand R akzeptiert, falls ein akzeptierender Zustand von F in R ist





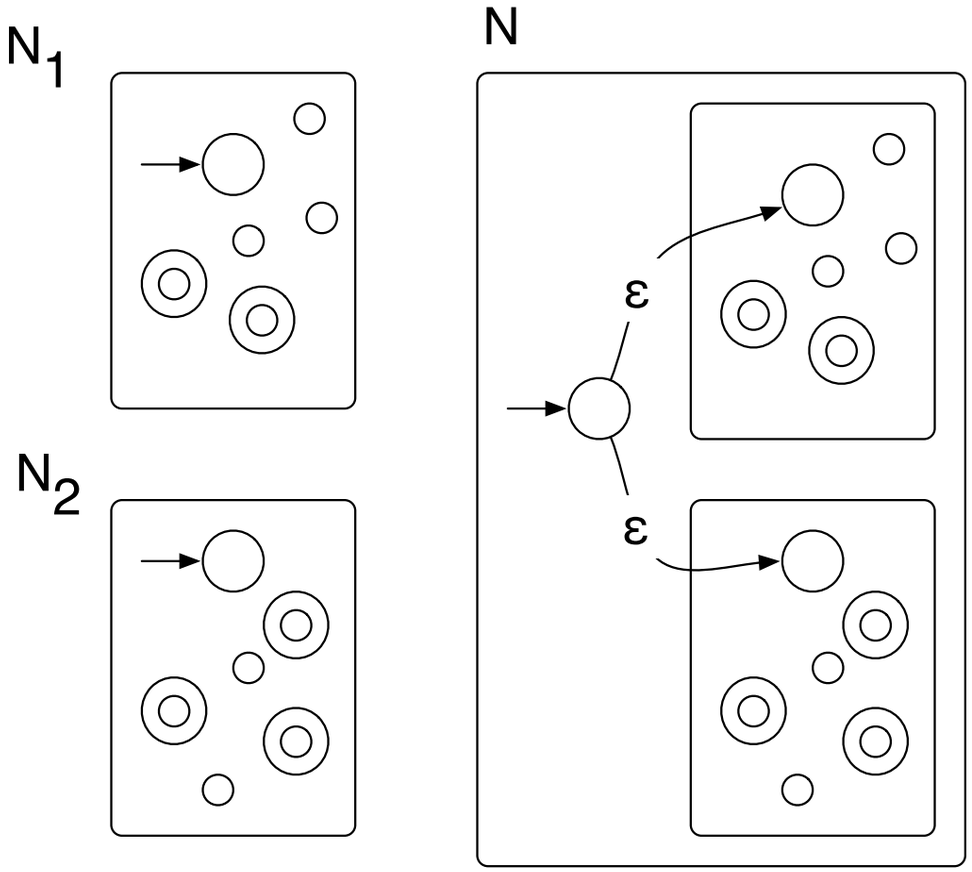
Ein alternativer Beweis für den Abschluß über der Vereinigung

- **Beweisskizze:**
  - Betrachte NFAs N_1 und N_2
  - Konstruiere N mit neuem Startzustand und ε-Übergangen zu den Startzuständen von N_1 und N_2
  - NFA N akzeptiert $L(N_1) \cup L(N_2)$



Abschluss unter Vereinigung

- **Alternativer Beweis**
  - Gegeben seien nichtdeterministische endliche Automaten
    - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ und
    - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$



- **Konstruktion von $N = (Q, \Sigma, \delta, q_0, F)$**
  - so dass $L(N) = L(N_1) \cup L(N_2)$
  - Zustandsmenge: $Q = \{q_0\} \cup Q_1 \cup Q_2$
  - Anfangszustand: q_0
  - Akzeptierende Zustände: $F = F_1 \cup F_2$
  - Übergangsfunktion

$$\delta(q,a) = \begin{cases} \delta_1(q,a)\ , & q \in Q_1 \\ \delta_2(q,a)\ , & q \in Q_2 \\ \{q_1, q_2\}\ , & q = q_0 \text{ und } a = \epsilon \\ \emptyset\ , & q = q_0 \text{ und } a \neq \epsilon \end{cases}$$

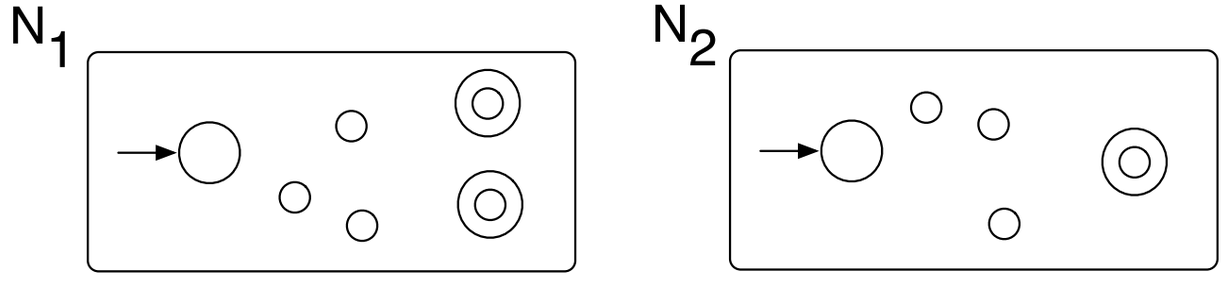
Formale Sprachen und
Endliche Automaten

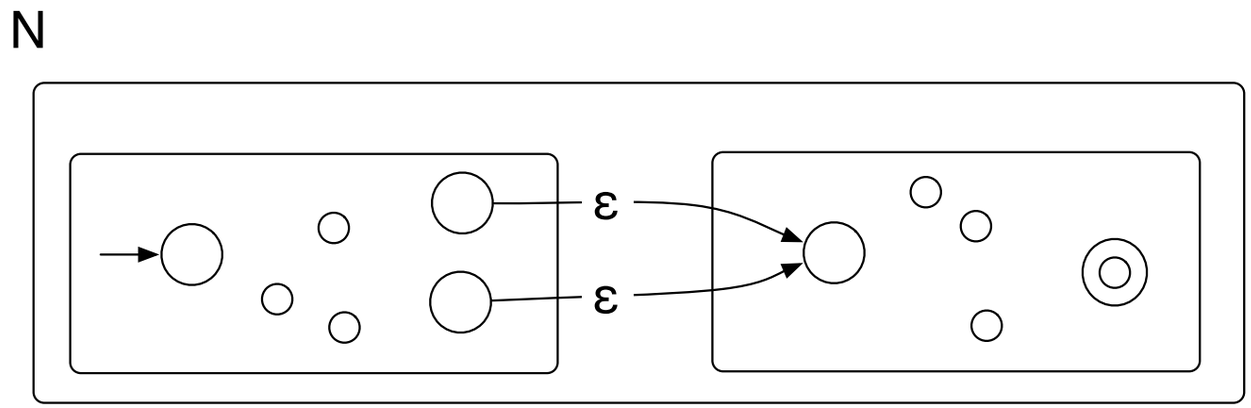
REG und reguläre Ausdrücke

56

Die regulären Sprachen sind gegenüber der Konkatenation abgeschlossen

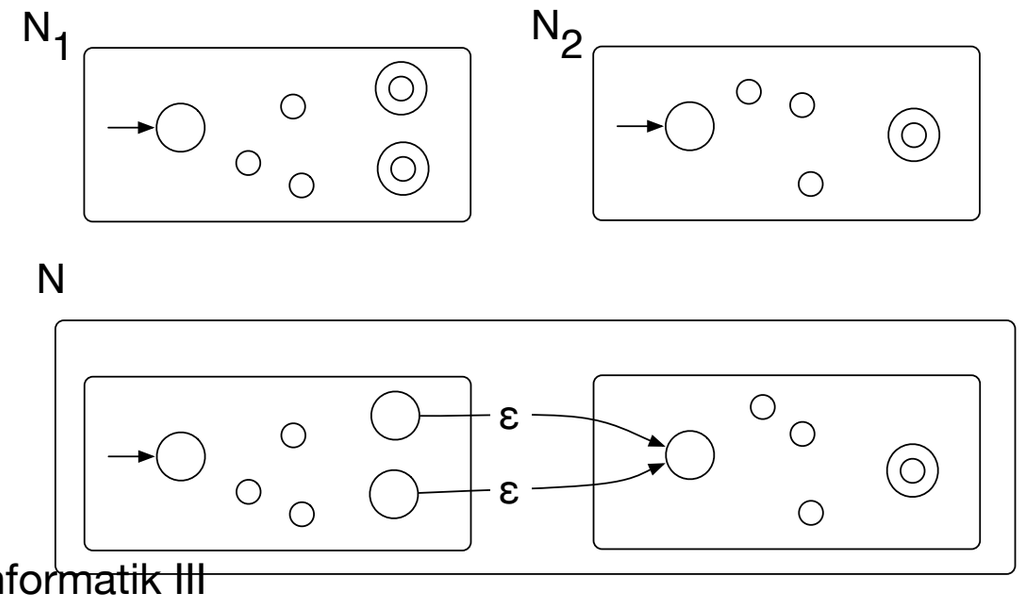
- **Beweisskizze:**
  - Betrachte NFA N_1 und N_2
  - Konstruiere NFA N mit ε-Übergangen von allen akzeptierenden Zuständen von NFA N_1 zu dem Startzustand von NFA N_2
  - Neuer Startzustand von N ist Startzustand von N_1
  - Die neuen akzeptierenden Zustände sind die von N_2
  - NFA N akzeptiert $L(N_1)L(N_2)$





Abschluss unter Konkatenation

- **Theorem**
  - Die Menge der regulären Sprachen ist abgeschlossen unter Konkatenation
- **Beweis:**
  - Gegeben seien nichtdeterministische endliche Automaten
    - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ und
    - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$



- **Konstruktion von N,**
  - sodass $L(N) = L(N_1)\, L(N_2)$
  - Zustandsmenge: $Q' = Q_1 \cup Q_2$
  - Anfangszustand: $q_0 = q_1$
  - Akzeptierende Zustände: $F = F_2$
- **Übergangsfunktion**

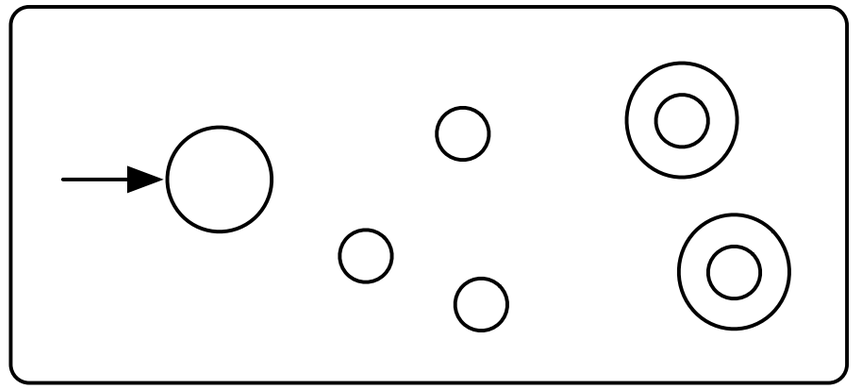
$$\delta(q,a) = \begin{cases} \delta_1(q,a)\ , & q \in Q_1 \text{ und } q \notin F_1 \\ \delta_1(q,a)\ , & q \in F_1 \text{ und } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\}\ , & q \in F_1 \text{ und } a = \epsilon \\ \delta_2(q,a)\ , & q \in Q_2 \end{cases}$$

- **Korrektheit der Konstruktion zur Übung empfohlen**

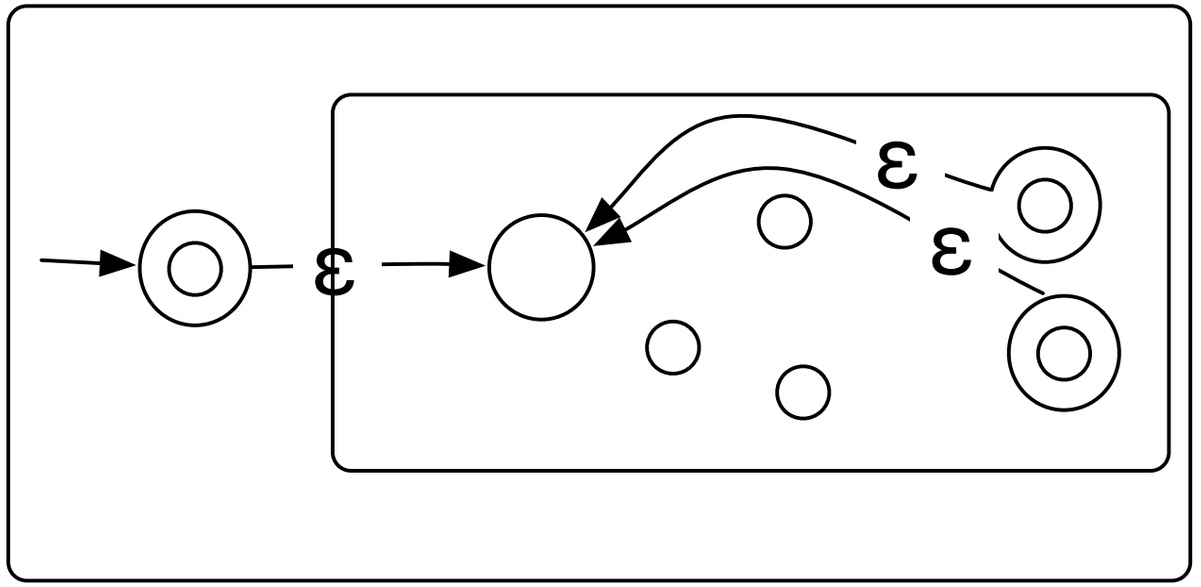
Die regulären Sprachen sind unter dem Stern-Operator abgeschlossen

- **Beweisskizze:**
  - Betrachte NFA N_1
  - Konstruiere NFA N mit neuem Startzustand
  - ε-Übergang vom neuem Startzustand zum Alten
  - ε-Übergangen von allen akzeptierenden Zuständen zum Exstartzustand
  - Der Rest von V bleibt gleich
  - $L(N) = L(N_1)^*$

N_1



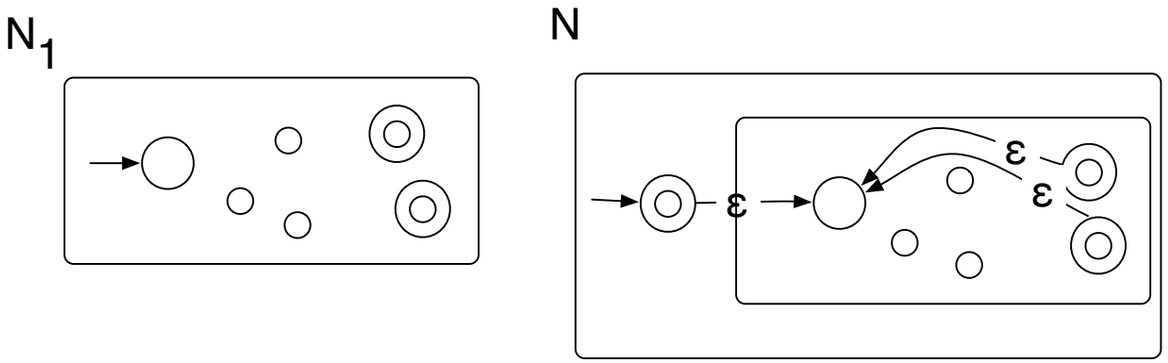
N



Abschluss unter der Stern-Operation

- **Theorem**
  - Die Menge der regulären Sprachen ist abgeschlossen unter der Stern-Operation

- **Beweis:**
  - Gegeben sei der nichtdeterministische endliche Automat
    - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ und



- **Konstruktion von N mit $L(N) = L(N_1)^*$**
  - Zustandsmenge: $Q' = \{q_0\} \cup Q_1$
  - Anfangszustand: q_0
  - Akzeptierende Zustände: $F = \{q_0\} \cup F_1$
  - Übergangsfunktion

$$\delta(q,a) = \begin{cases} \delta_1(q,a) , & q \in Q_1 \text{ und } q \notin F_1 \\ \delta_1(q,a) , & q \in F_1 \text{ und } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} , & q \in F_1 \text{ und } a = \epsilon \\ \{q_1\} , & q = q_0 \text{ und } a = \epsilon \\ \emptyset , & q = q_0 \text{ und } a \neq \epsilon \end{cases}$$

  - Rest des Beweises zum Üben

Reguläre Ausdrücke

- **Definition**
- **R ist ein regulärer Ausdruck, wenn R einer der folgenden Darstellungen besitzt**
  - a, für ein Zeichen $a \in \Sigma$
  - ε
  - $\emptyset$
  - $(R_1 \cup R_2)$, wenn R_1 und R_2 reguläre Ausdrücke sind
  - $(R_1 \circ R_2)$, wenn R_1 und R_2 reguläre Ausdrücke sind
  - $(R_1)^*$, wenn R_1 ein regulärer Ausdruck ist

Notation

- **Statt R1 ° R2 schreiben wir R1 R2**
- **Bindung:**
  - zuerst Stern, dann Konkatenation, dann Vereinigung

    $a \circ b \cup i^* \circ a^* \cup c^* \circ k$

    = (a ∘ b) ∪ ((i*) ∘ a*) ∪ ((c*) ∘ k)
  - Schöner:

    = ab ∪ i*a* ∪ c*k

Beispiele

- **Zum Warmwerden: Was ist das?**
  - `flick ∪ flack = fl (i ∪ a) ck`
  - `fidera(la)*la`
  - `0*10*`
  - `(0∪1∪2∪3∪4∪5∪6∪7∪8∪9)*(0∪5)`
- **Kniffliger:**
  - otto ∪ Ø
  - otto ε
  - oεtεtεo ○ Ø
- **Praktisch unlösbar (oder?)**
  - ε ∪ Ø
  - ε ○ Ø
  - ε*
  - Ø*
- **Der Knüller**
  - Ø ○ (Ø ∪ (Ø ○ Ø))*

Reguläre Ausdrücke beschreiben reguläre Sprachen

- **Lemma**
  - Jeder reguläre Ausdruck R beschreibt eine reguläre Sprache
- **Beweis**
  - Wir konvertieren R in einen NFA

1. Fall R = a, für a ∈ Σ
   - Automat:

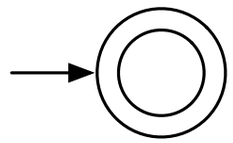
   - Formal:

$$N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$$

$$\delta(q_1, a) = \{q_2\}$$

$$\delta(r, b) = \emptyset, \text{ für } r \neq q_1 \text{ oder } b \neq a$$

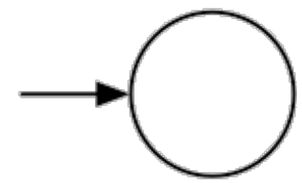
2. Fall R = ε
   - Automat:



   - Formal: $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$

$$\delta(r, b) = \emptyset, \text{für alle } r, b$$

3. Fall R = Ø
   - Automat:



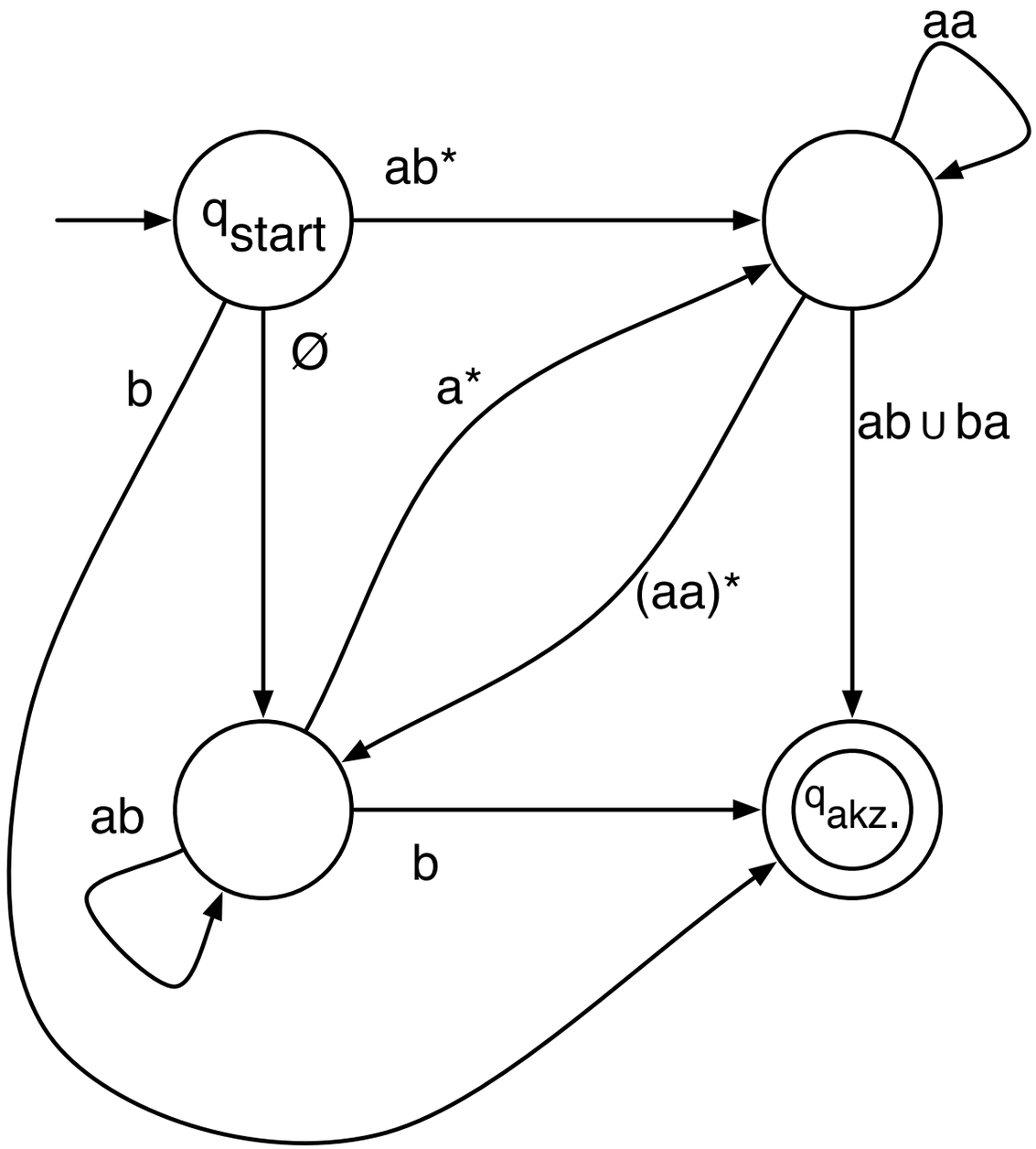
   - Formal: $N = (\{q\}, \Sigma, \delta, q, \emptyset)$

$$\delta(r, b) = \emptyset, \text{ für alle } r, b$$

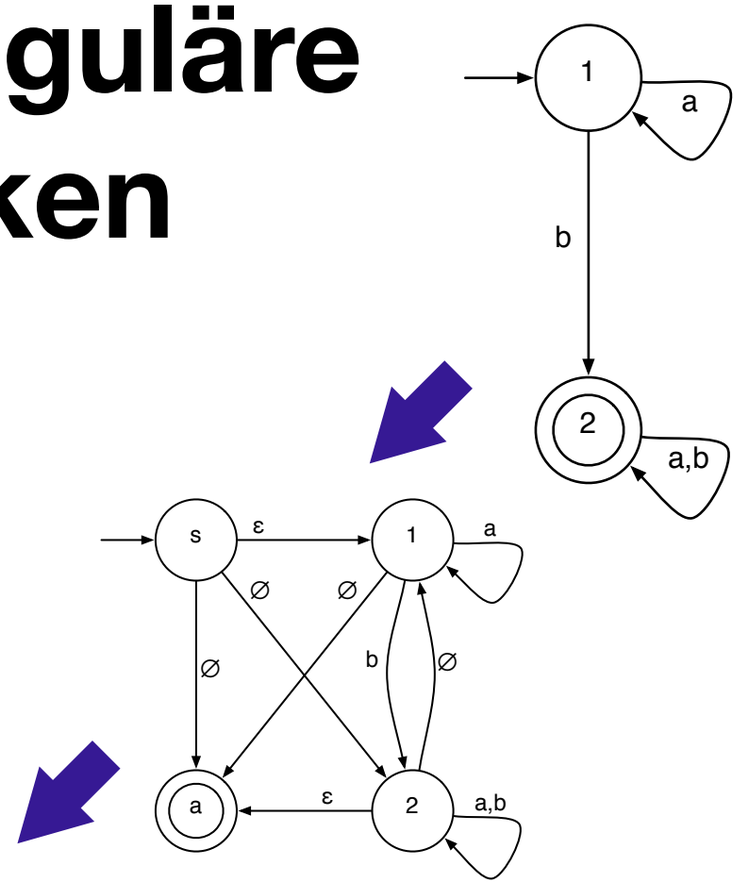
4. Fall: R = ($R_1 \cup R_2$)
5. Fall: ($R_1 \circ R_2$)
6. Fall: (R_1)* siehe vorherige Folien

REG lässt sich durch reguläre Ausdrücke ausdrücken

- **Strategie:**
  - Einführung der verallgemeinerten nichtdeterministischen endlichen Automaten (Generalized Non-deterministic Finite Automata - GNFA)
  - NFA → GNFA
  - GNFA → Regulärer Ausdruck
- **Eigenschaften GNFA:**
  - GNFA = NFA + reguläre Ausdrücke
  - Reguläre Ausdrücke auf Übergängen
  - Ein akzeptierender Zustand
  - Alle Übergänge existieren außer
    - Kein Übergang zum Startzustand
    - Kein Übergang vom akz. Zustand



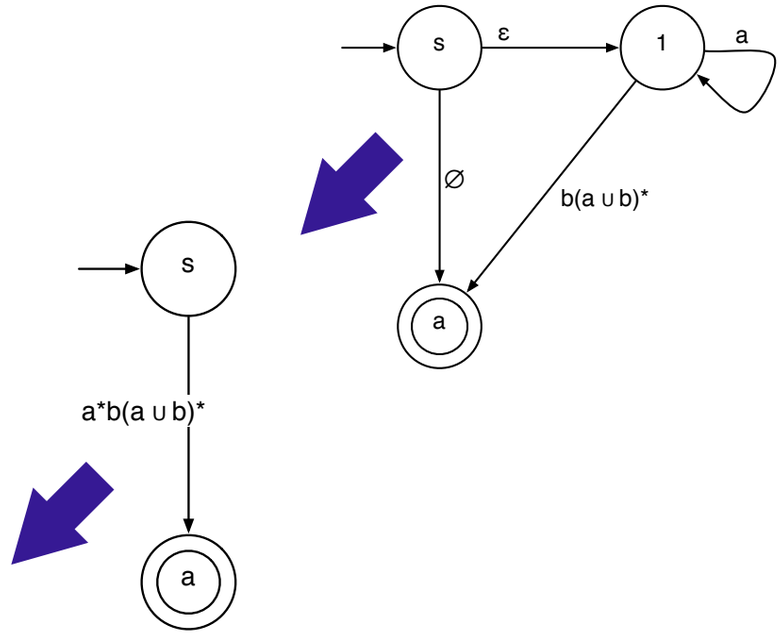
REG lässt sich durch reguläre Ausdrücke ausdrücken



- **Strategie:**
  - NFA mit k Zuständen
    → GNFA mit k+2 Zuständen
  - GNFA mit k+2 Zuständen
    → GNFA mit k+1 Zuständen
  - GNFA mit k+1 Zuständen
    → GNFA mit k Zuständen

...

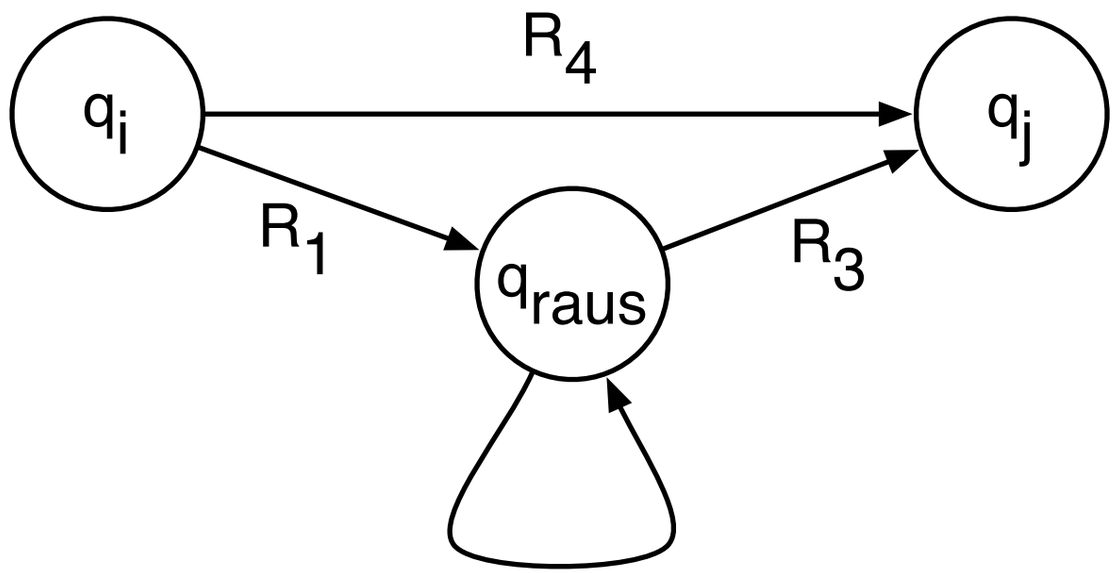
- GNFA mit 3 Zuständen
  → GNFA mit 2 Zuständen
- GNFA mit 2 Zuständen
  → Regulärer Ausdruck



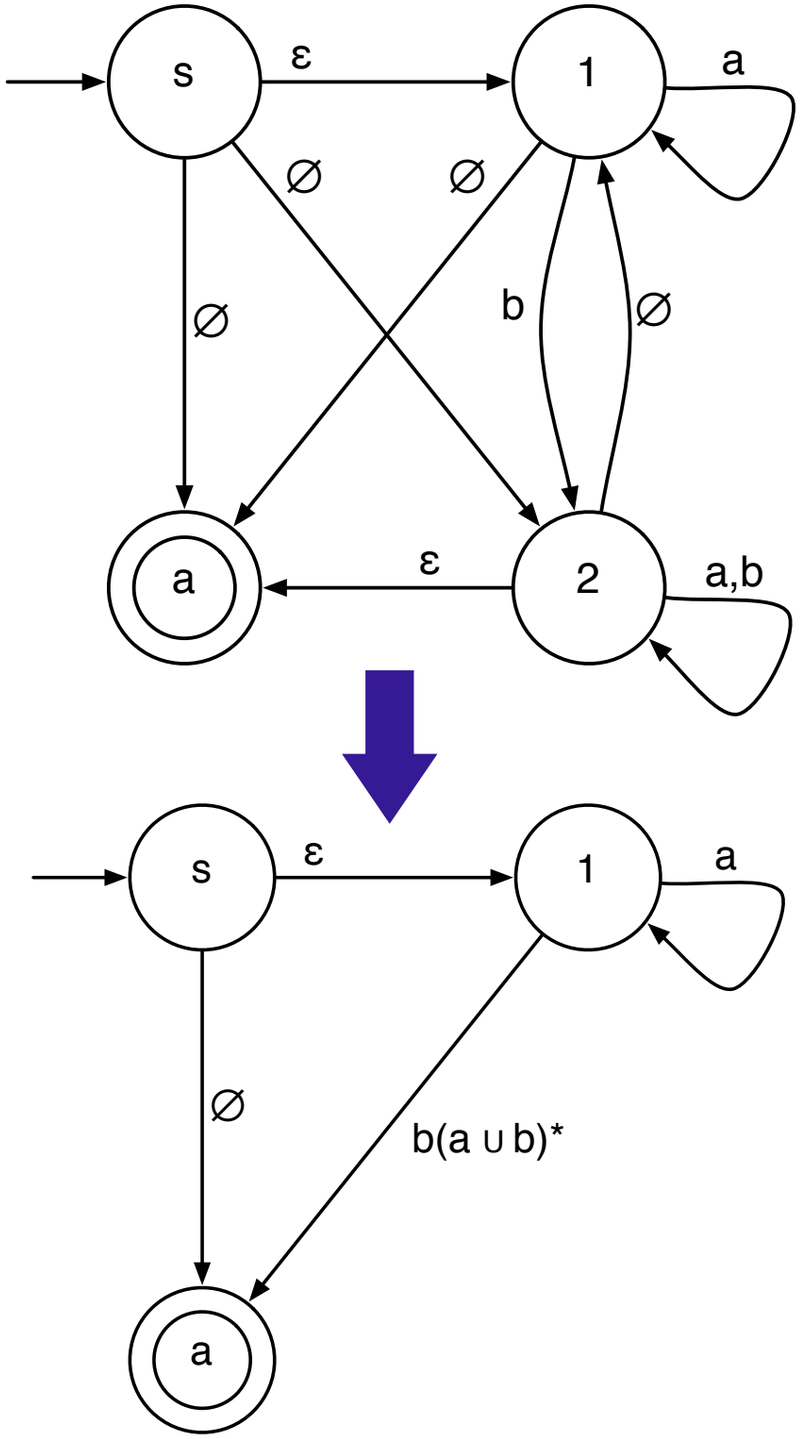
`a*b(a ∪ b)*`

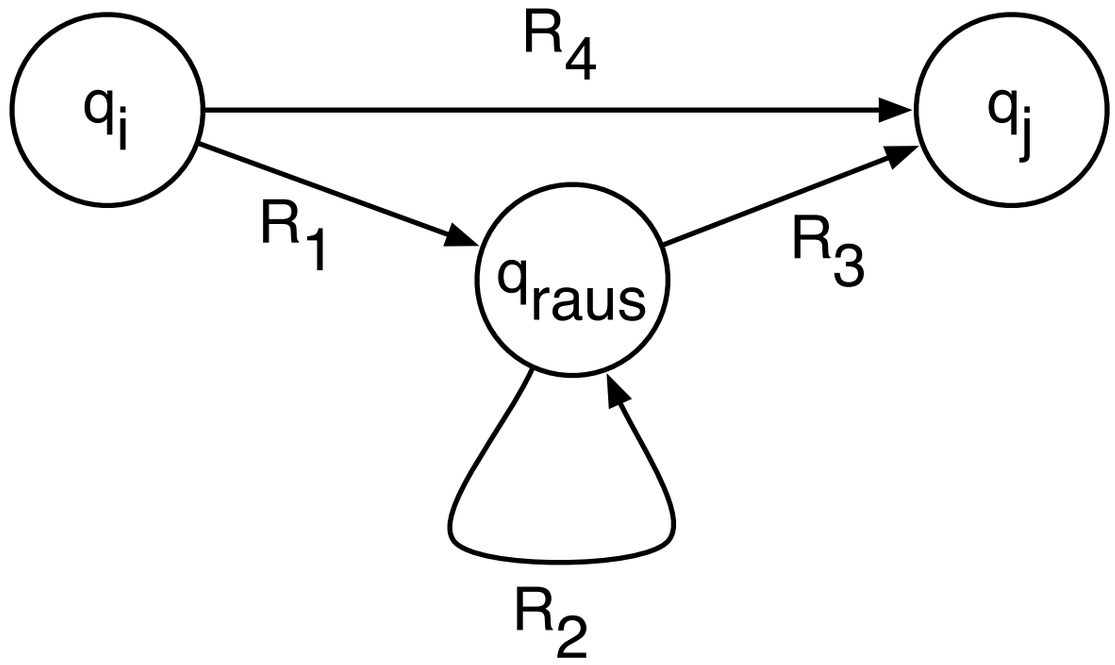
GNFA mit k Zuständen
→ GNFA mit k-1 Zuständen

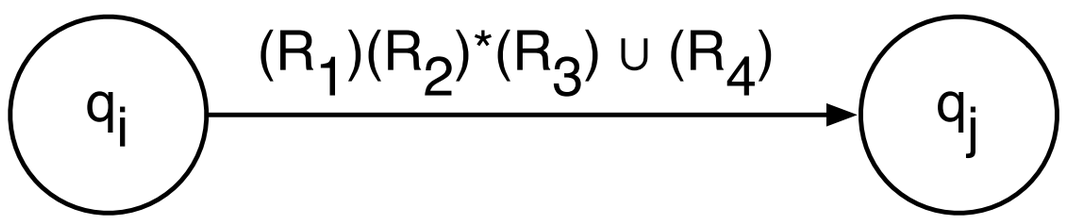
- **Wie kann man einen Zustand im GNFA einsparen?**
  - Zustand q_{raus} soll raus
  - Betrachte alle anderen Paare q_i,q_j
  - Jeder Weg von q_i nach q_j kann entweder
    - nach q_{raus} führen (R_1)
    - dort beliebig häufig q_{raus} die Schleife über q_{raus} nehmen $(R_2)^*$
    - dann nach q_j gehen (R_3)
  - oder überhaupt nicht über q_{raus} gehen:
    - (R4)



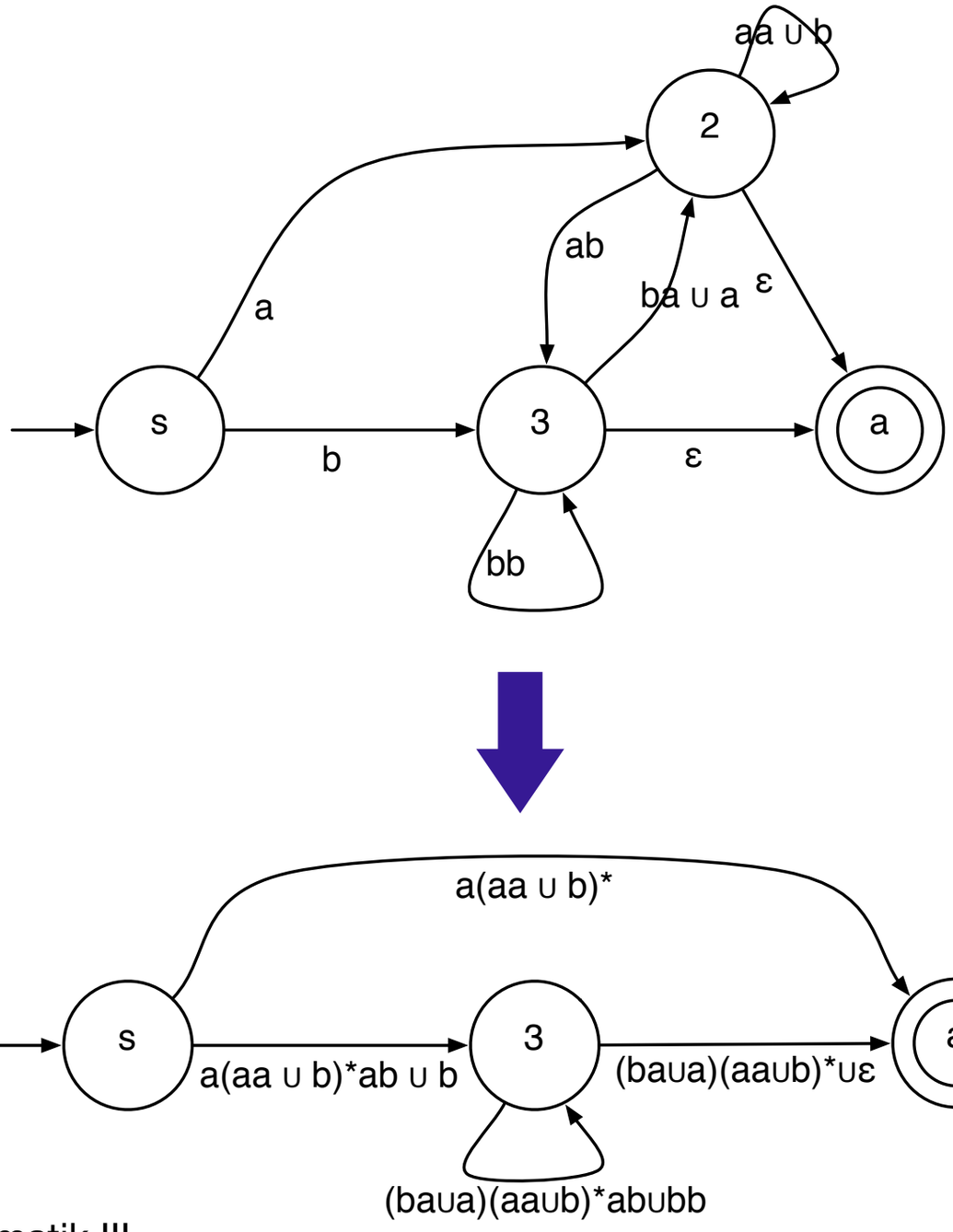
GNFA mit k Zuständen
→ GNFA mit k-1 Zuständen

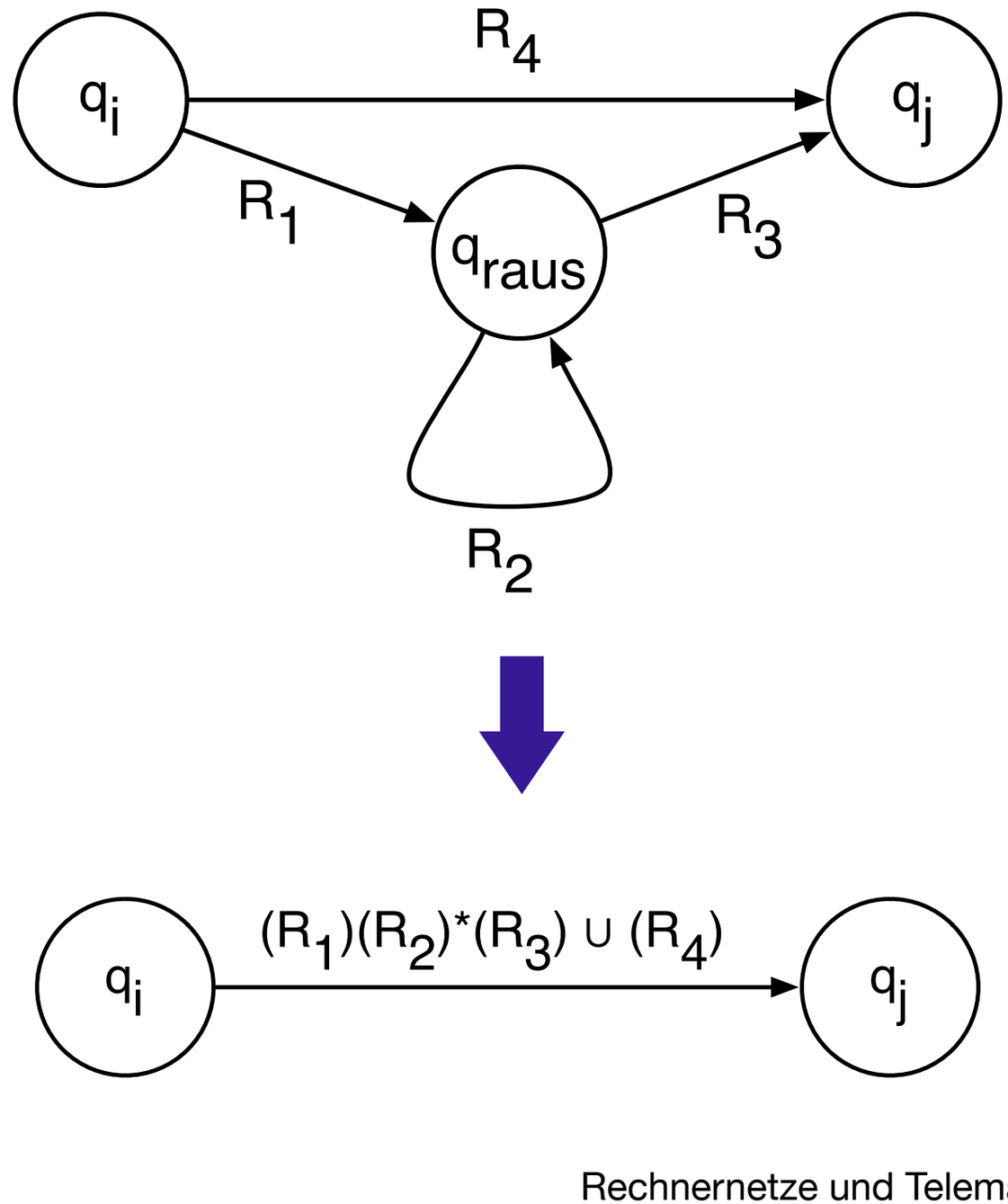






GNFA mit k Zuständen
→ GNFA mit k-1 Zuständen



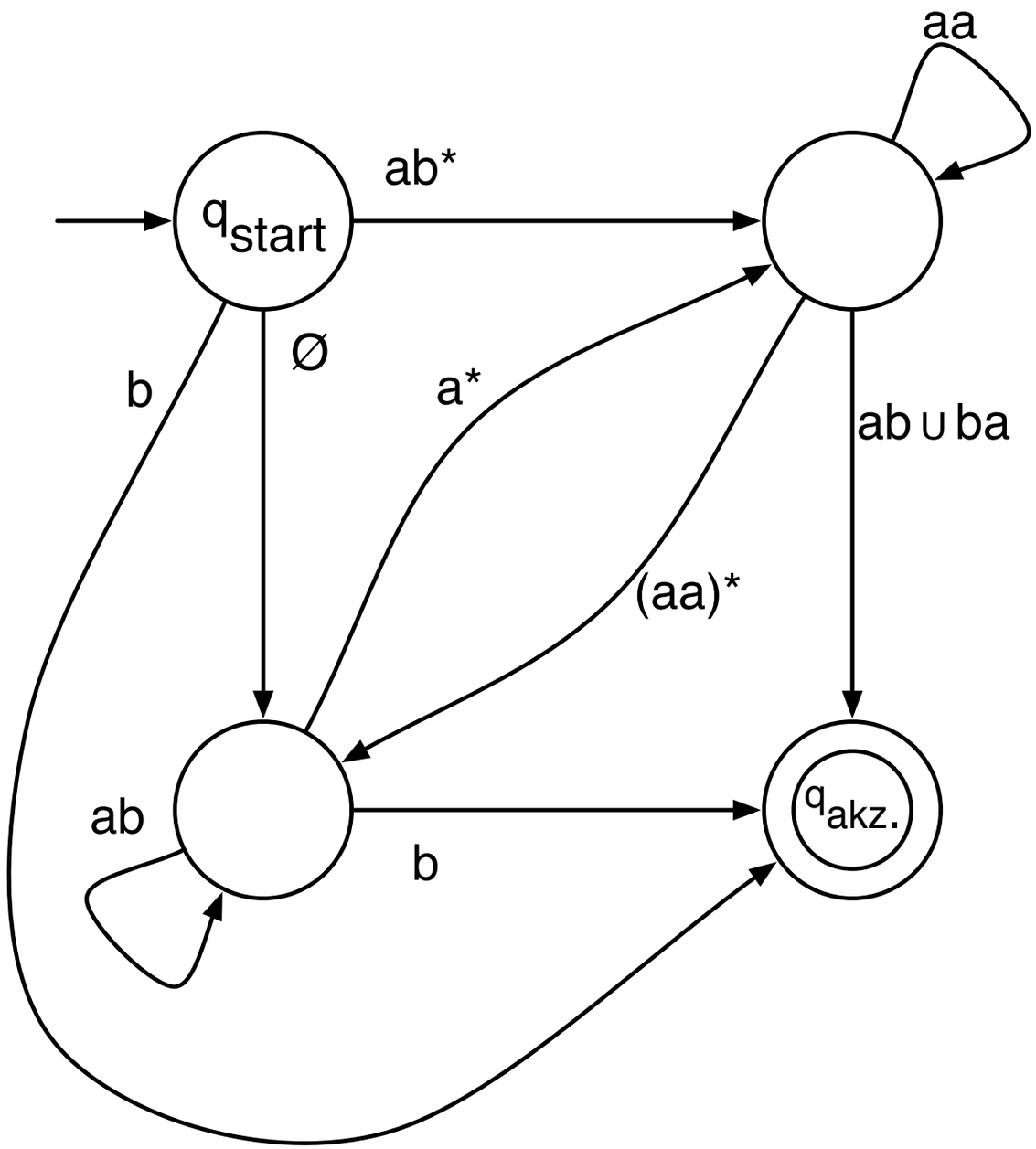


Und jetzt ausführlich:

- **Lemma**
  - Jeder GNFA mit k>2 Zuständen läßt sich in einem äquivalenten mit k-1 Zuständen umformen.
- **Beweisschritte**
  - Formale Definition des GNFA
  - Formale Definition der Berechung eines GNFAs
  - Definition der Operation Konvertiere(G)
    - der ein GNFA um einen Zustand reduziert
  - Beweis der Korrektheit der Operation

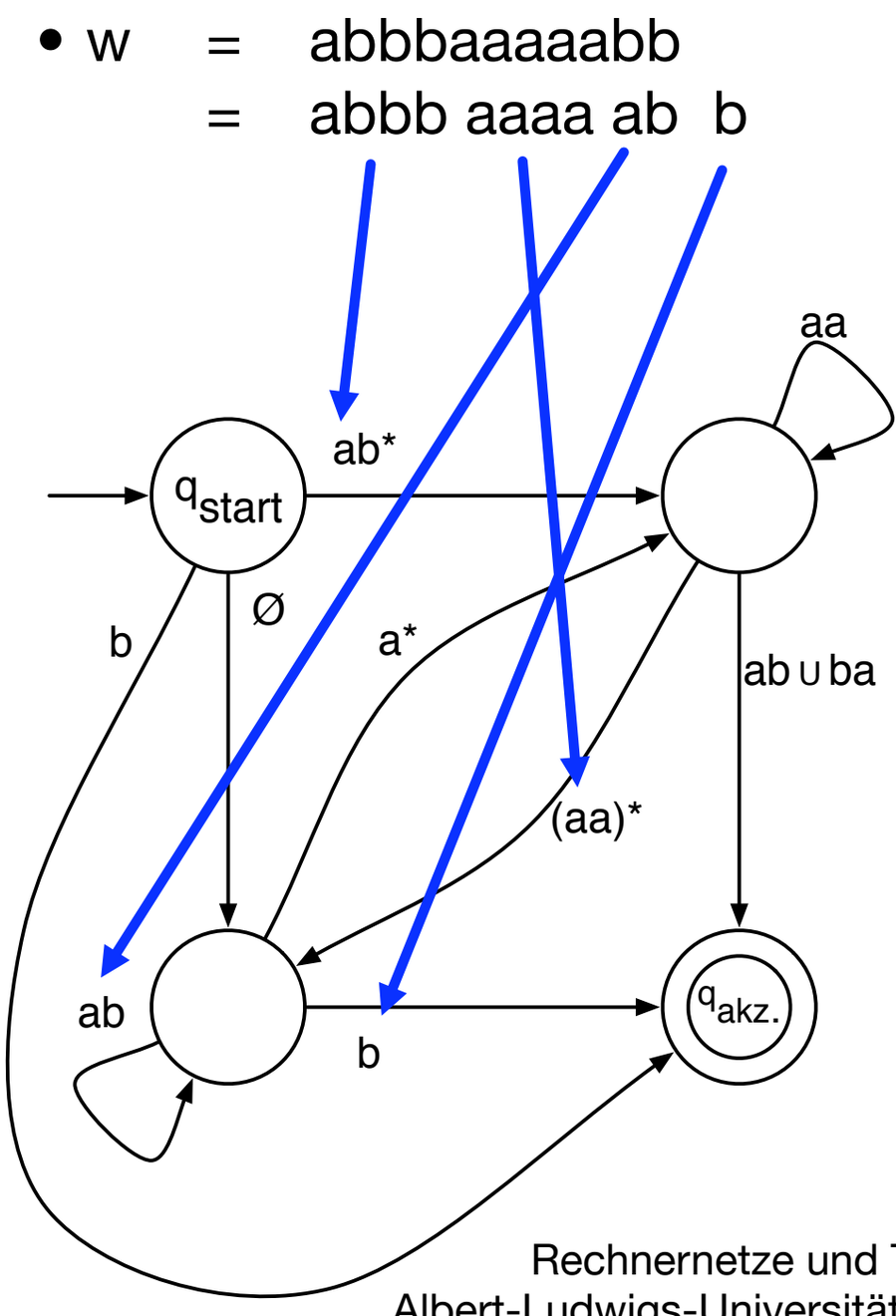
Definition eines GNFA

- **Definition**
  - Ein verallgemeinerter nichtdeterministischer endlicher Automat (GNFA) wird durch das 5-Tupel $(Q, \Sigma, \delta, q_{start}, q_{akz})$ beschrieben
    - Q: Endliche Menge von Zuständen
    - Σ: Alphabet
    - δ: $(Q \backslash \{q_{akz}\}) \times (Q \backslash \{q_{start}\}) \rightarrow R$ ist die Übergangsfunktion
      - R ist die Menge der regulären Ausdrücke
    - $q_{start} \in Q$: ist der Startzustand
    - $q_{akz} \in Q$: ist der akzeptierende Endzustand



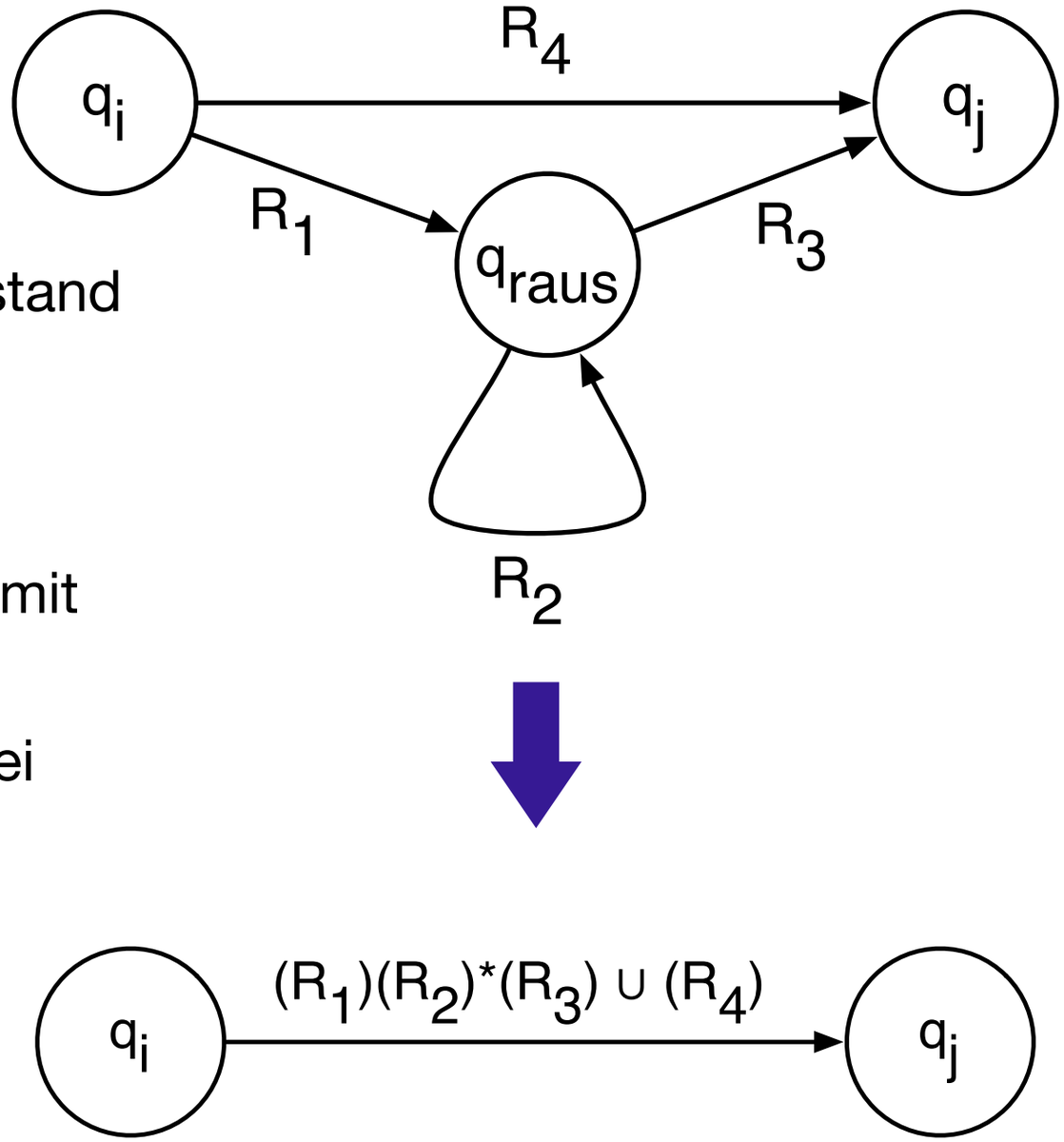
Berechnung eines GNFA

- **Definition**
  - Ein GNFA N = (Q, Σ, δ, q_{start}, q_{akz}) akzeptiert ein Wort w ∈ Σ* falls es
    - eine Darstellung der Eingabe w = w_1 w_2 ... w_m mit w ∈ Σ* gibt
    - und es eine Folge q_0 q_1 ... q_m von Zuständen aus Q gibt, wobei
      - $q_0 = q_{start}$
      - $q_m = q_{akz}$
      - für alle i: $w_i \in L(R_i)$ gilt $R_i = \delta(q_{i-1}, q_i)$
  - dann akzeptiert N das Wort.



Der KONVERTIERER

- **Konvertiere(G=(Q, Σ, δ, q_{start}, q_{akz}) :GNFA):**
  - k = Anzahl der Zustände von G
- **Falls k=2 dann**
  - Gib regulären Ausdruck zwischen Startzustand und akz. Zustand aus
- **Falls k>2 dann**
  - Wähle beliebig $q_{raus} \in Q \setminus \{q_{akz}, q_{start}\}$
  - Betrachte GNFA G' = (Q', Σ, δ', q_{start}, q_{akz}) mit
    - Q' = Q \ $\{q_{raus}\}$
    - Für alle $q_i \in Q \setminus \{q_{akz}\}$ und $q_j \in Q \setminus \{q_{start}\}$ sei $\delta'(q_i,q_j) = (R_1)\,(R_2)^*\,(R_3) \cup (R_4)$
    - wobei $R_1 = \delta(q_i,q_{raus})$, $R_2 = \delta(q_{raus},q_{raus})$, $R_3 = \delta(q_{raus},q_j)$, $R_4 = \delta(q_i,q_j)$
  - Berechne rekursiv Konvertiere(G')



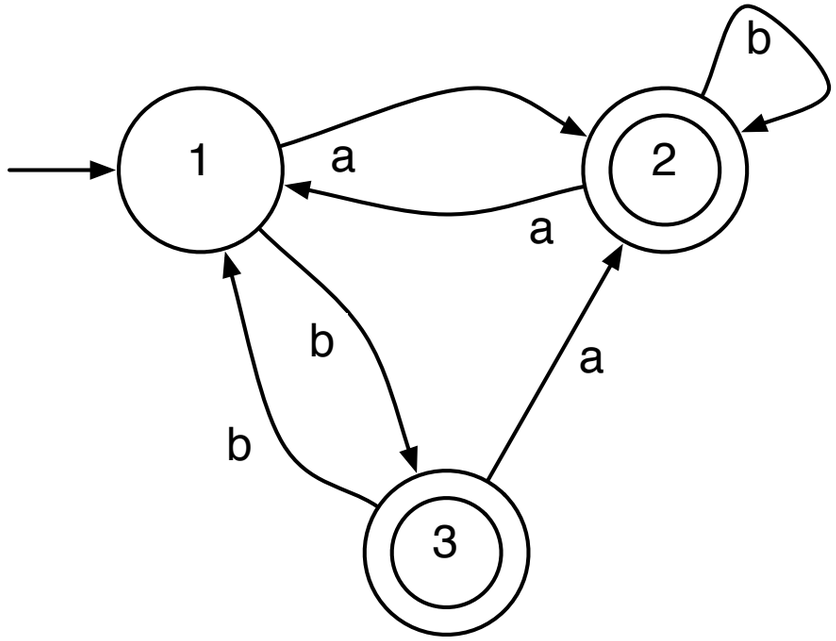
Beweis der Korrektheit

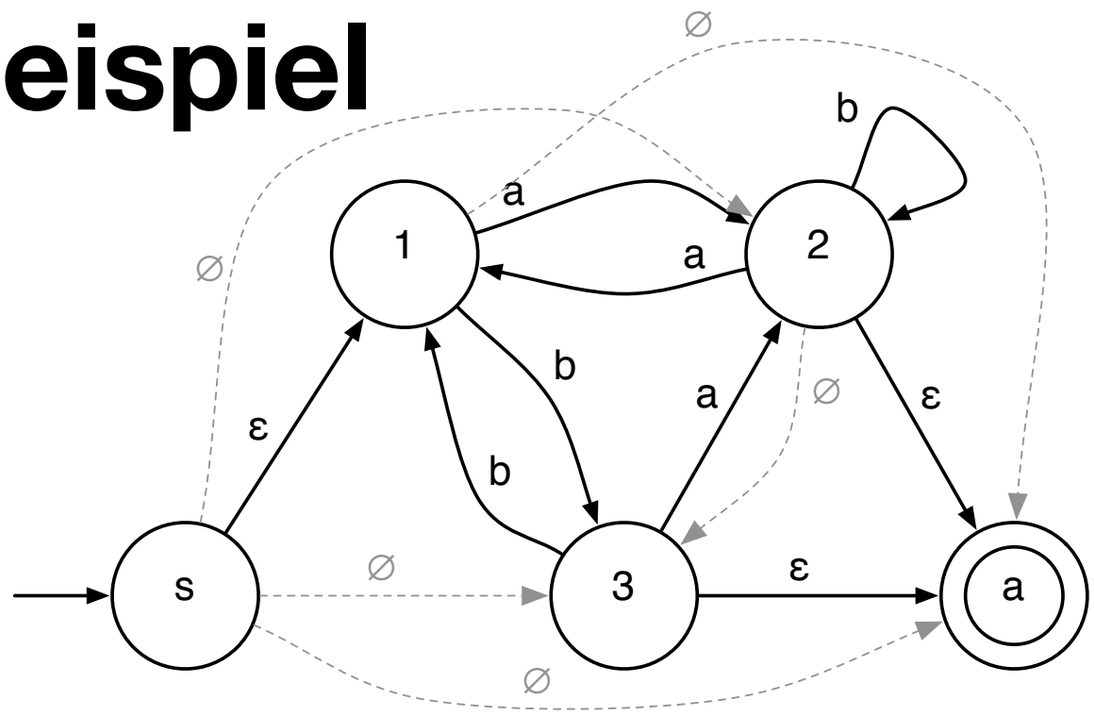
- **Behauptung**
  - Für jeden GNFA G ist
    G' = Konvertiere(G) äquivalent zu G.
- **Beweis durch Induktion über**
  k = Anzahl der Zustände von G
- **Basis (Anker): k=2**
  - Der GNFA G hat nur einen Übergang
  - Nach Definition ist er äquivalent mit dem regulären Ausdruck auf dem Übergang
- **Induktionsschritt: Angenommen die Behauptung ist wahr für k-1 Zustände**
  - Angenommen G akzeptiert w und sei q_{start}, q_1, q_2, ... q_{akz} die Folge der Zustände
  - 1. Fall: q_{raus} ist nicht in der Folge:
    - dann bleibt alles unverändert
  - 2. Fall: q_{raus} ist in der Folge:
    - Dann stehen vor oder nach einer Folge von q_{raus} andere Zustände
    - Seien dies q_i und q_j
    - dann ist das Teilwort von q_i nach q_j im Ausdruck $(R_1)\ (R_2)^*\ (R_3)$ enthalten
- **Angenommen G' akzeptiert w**
  - Dann sei q_{start}, q_1, q_2, ... q_{akz} die Folge der Zustände
  - Dann kann jeder Übergang mit Teilwort w' zwischen zwei Zuständen q_i und q_j dargestellt werden durch einen direkten Übergang in G falls w' $\in$ L(R_4)
  - oder durch einen Weg über q_{raus} falls w' $\in$ L((R1) (R2)* (R3))
  - In jedem Fall würde auch G das Wort w akzeptieren
- **G akzeptiert also gdw. G' akzeptiert**

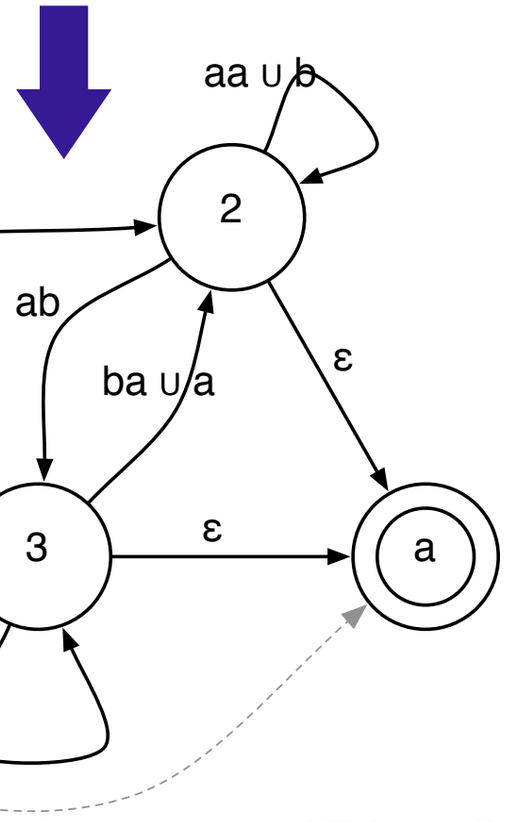
Daraus folgt...

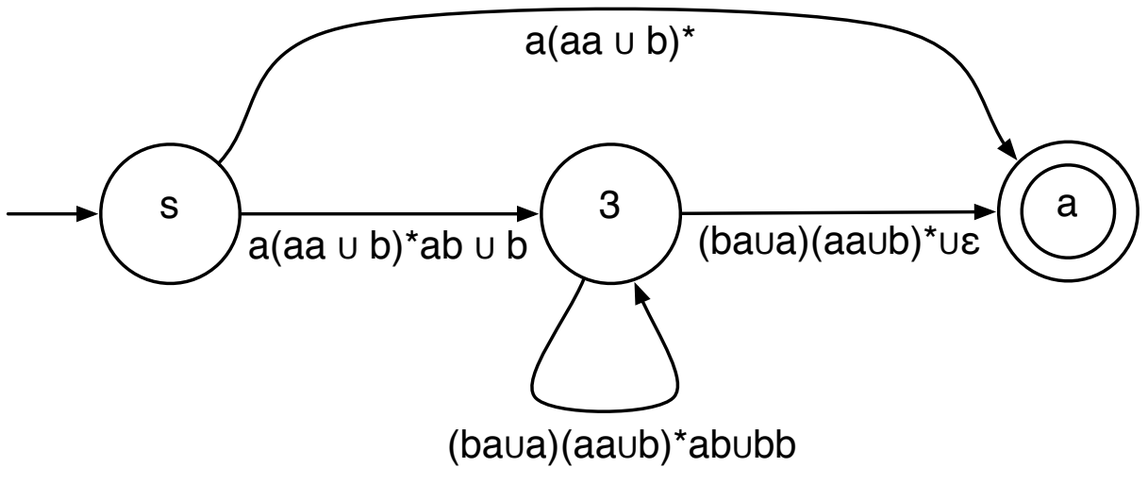
- **Theorem**
  - Die regulären Ausdrücke beschreiben genau die regulären Sprachen.

Noch ein Beispiel

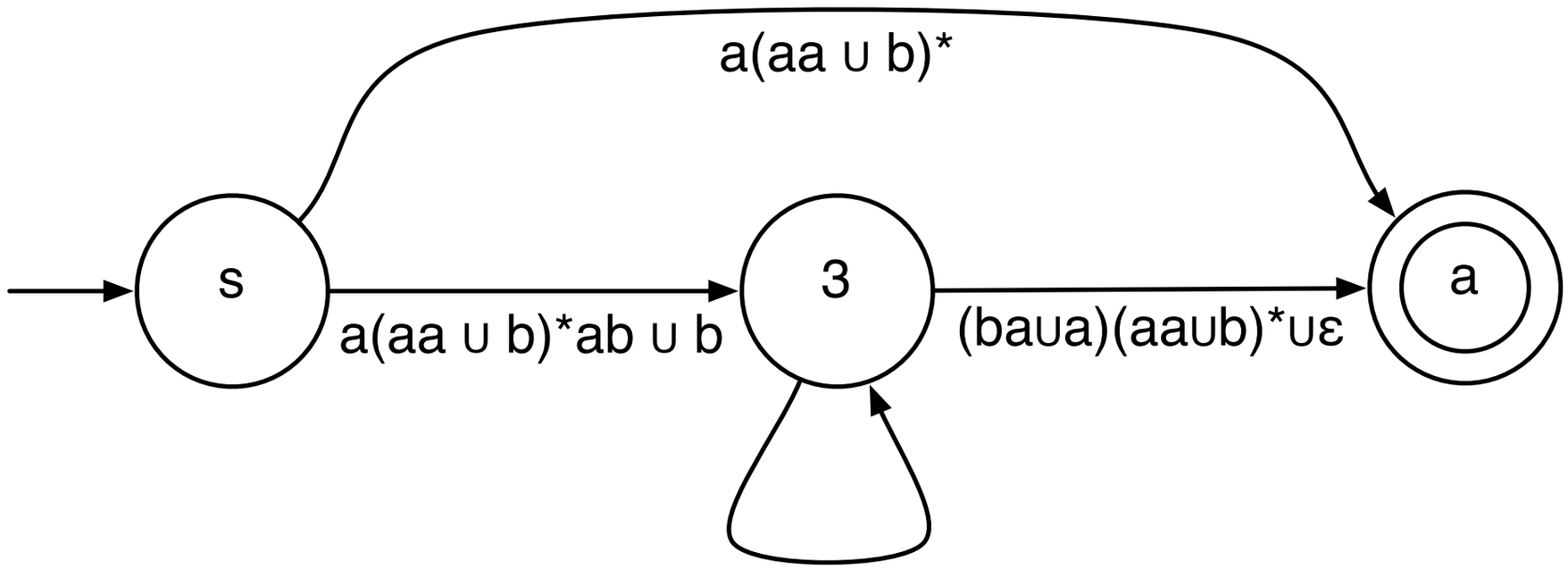




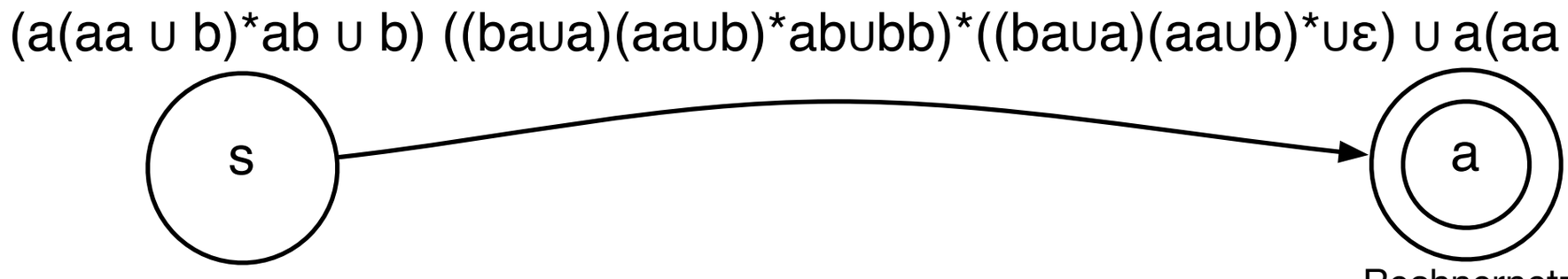




Weiter



(ba∪a)(aa∪b)*ab∪bb



Formale Sprachen und
Endliche Automaten

Pumping-Lemma für reguläre Sprachen

78

Das Pumping-Lemma

- **Motivation**
  - Reguläre Sprachen sind die Stottottottottererer der Sprachen
  - Was ist nicht regulär?
    - Palindrom $= \{w \in \{a,b\}^* \mid \forall i \in \{1,\ldots,|w|\} : w_i = w_{|w|-i+1}\}$
      $= \{\epsilon, a, b, aa, bb, aaa, aba, bab, bbb, \ldots\}$
    - Kopie = { w w | w $\in \Sigma^*$}
    - Klammersprachen, z.B. Syntaktische Überprüfung von
      `((a+b)-b = (a))/(b))+b-a`
    - Zählsprachen, z.B.
      { w $\in$ {0,1}* | die Anzahl der 0er = Anzahl der 1er}
    - Menge der Primzahlen, Quadratzahlen
- **Wie kann man aber zeigen, dass etwas nicht regulär ist?**
  - Durch das **Pumping-Lemma**

Das Pumping-Lemma

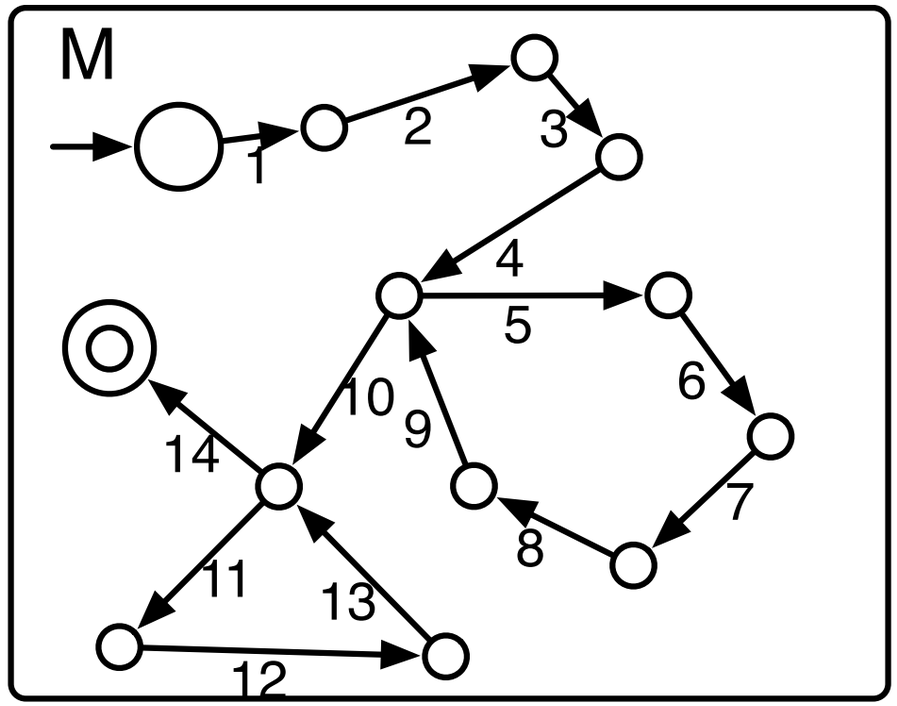
- **Pumping-Lemma**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl p>0
    - so dass für jedes Wort $s \in L$ mit $|s| \geq p$
    - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
      - für alle $i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.

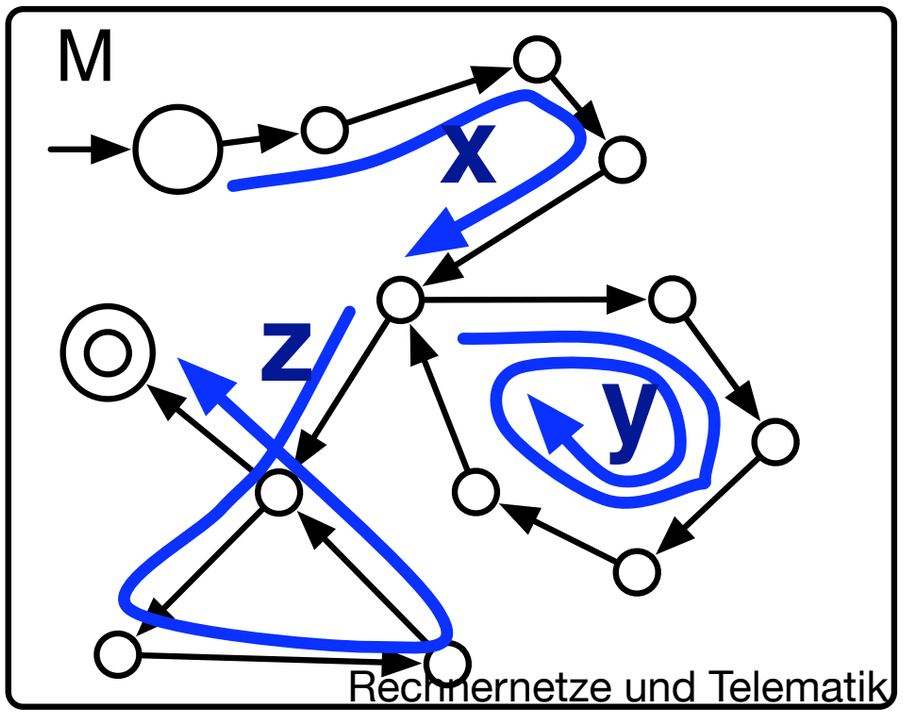
- **Positive Aussage:**
  - Betrachte Sprache:
    - A = L(Stott(ott)*(er)*)
    - p = 8
    - s = Stottotterererererer
    - x=Stott, y=ott
    - z=erererererer
    - $y \neq \varepsilon$
    - $|xy| \leq 8$
  - All diese Worte sind in A:
    - Stotterererererer
    - Stottotterererererer
    - Stottottotterererererer

Pumping-Lemma
Beweisidee

- **Pumping-Lemma**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl p>0
    - so dass für jedes Wort $s \in L$ mit $|s| \geq p$
    - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
      - $\forall\ i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.

- **Beweisidee**
  - Es gibt nur endlich viele Zustände im DFA von A
  - Auf langen Worten wiederholen sich manche Zustände
  - Das zugehörige Wort y kann also beliebig oft eingesetzt werden!





Der Beweis des Pumping-Lemmas

- **Pumping-Lemma**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl p>0
    - so dass für jedes Wort s ∈ L mit |s|≥p
    - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
      - $\forall i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.
- **Beweis:**
  - Sei M= (Q, Σ, δ, q_0, F) ein DFA der A akzeptiert mit p=|Q| Zuständen
  - Sei s = $s_1s_2...s_n$ ein Wort in A der Länge n≥p
  - Sei q = $r_1r_2...$ r_{n+1} die Folge der Zustände in M bei der Berechnung von s.
  - Diese Folge hat Länge n+1 ≥ p+1
  - Dann muss ein Zustand mehr als einmal vorkommen in den ersten p+1 Zuständen
  - Sei r_j das erste Vorkommen solch eines Zustandes und sei r_k das zweite Vorkommen
  - damit ist k ≤ p+1
  - Sei x = $s_1s_2...s_{j-1}$, y = $s_js_{j+1}...s_{k-1}$, z = $s_ks_{k+1}...s_n$
  - Dann muss M xy^iz akzeptieren, da
    - x Startzustand r_1 in Zustand r_j,
    - y Zustand r_j in r_j und
    - z Zustand r_j in den akz. Zustand r_n
  - überführt.

Das Pumping-Lemma ist ein Killerargument

- **Beispiel:**
  - Sei $B = \{0^n1^n \mid n \geq 0\}$
- **zu zeigen: B ist nicht regulär**
  - Angenommen doch
  - (Pumping-Lemma)
  - Dann gibt es eine Zahl p>0
  - so dass für jedes Wort $s \in L$ mit $|s| \geq p$
  - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
    - für alle $i \geq 0$: $xy^iz \in B$
    - $|y| > 0$
    - $|xy| \leq p$.
- **Daraus folgt für $s = 0^p1^p \in B$**
  - dann ist $xy \in L(0^*)$
  - und $|y| > 0$
  - Sei $m = |y|$
- **Dann müssten nach dem Pumping-Lemma folgende Worte in B sein:**
  - i=0: Das Wort $xz = 0^{p-m}1^p$
  - i=1: Das Wort $xyz = 0^p1^p$
  - i=2: Das Wort $xyyz = 0^{p+m}1^p$
  - ...
  - Bis auf i=1 gilt $xy^iz \notin B$
- **Das Pumping-Lemma liefert Worte, die nicht in B sind**
- **Daher kann B nicht regulär sein**

Das Pumping-Lemma noch einmal

- **Betrachte die Sprache**
  - $F = \{ww \mid w \in \{0,1\}^*\}$
- **zu zeigen: B ist nicht regulär**
  - Angenommen doch
  - (Pumping-Lemma)
  - Dann gibt es eine Zahl $p>0$
  - so dass für jedes Wort $s \in L$ mit $|s| \geq p$
  - s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
    - für alle $i \geq 0$: $xy^iz \in B$
    - $|y| > 0$
    - $|xy| \leq p$.
- **Daraus folgt für $s = 0^p10^p1 \in F$**
  - dann ist $xy \in L(0^*)$
  - und $|y| > 0$
  - Sei $m = |y|$
- **Dann müssten nach dem Pumping-Lemma folgende Worte in C sein:**
  - i=0: Das Wort $xz = 0^{p-m}10^p1$
  - i=1: Das Wort $xyz = 0^p10^p1$
  - i=2: Das Wort $xyyz = 0^{p+m}10^p1$
  - i=3: Das Wort $xy^3z = 0^{p+2m}10^p1$
  - ...
  - Bis auf i=1 gilt $xy^iz \notin F$
  - Das Pumping-Lemma liefert Worte, die nicht in F sind
- **Daher kann F nicht regulär sein**

Das Pumping-Lemma, das 3. Mal

- **Betrachte die Sprache**
  - $D = \{1^m \mid \text{für } m=n^2, n\geq 0\}$
- **zu zeigen: B ist nicht regulär**
  - Angenommen doch
  - (Pumping-Lemma)
  - Dann gibt es eine Zahl $p>0$
  - so dass für jedes Wort $s \in L$ mit $|s|\geq p$
  - s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
    - für alle $i\geq 0$: $xy^iz \in B$
    - $|y| > 0$
    - $|xy| \leq p$.
- **Betrachte 1^m**
  - mit $m\geq p$ für ein $p>1$
- **Dann ist**
  - $|xz| = m-p$ eine Quadratzahl
  - $|xyz| = m$ eine Quadratzahl
  - $|xy^2z| = m+p$ eine Quadratzahl
  - $|xy^3z| = m+2p$ eine Quadratzahl
  - ...
- **Aber: der Abstand zwischen Quadratzahlen wächst**
  - $(n+1)^2-n^2 = 2n+1$
  - Also ist irgendwann $2n+1 > p$ und dann kann für $n>(p-1)/2$
  - nicht zugleich $|xyz|=n^2$ und $|xy^2z|$ Quadratzahlen sein.
- **Also ist D nicht regulär**

Formale Sprachen und
Endliche Automaten

Äquivalenzklassen und der Satz von Myhill-Nerode

Äquivalenzklassen
Definition und Beispiel

- **Definition**
  - Für eine Sprache $L \subseteq \Sigma^*$ bezeichnen wir zwei Worte $x,y \in \Sigma^*$ als L-äquivalent, geschrieben als
    $$x \equiv_L y,$$
    wenn für alle Worte $z \in \Sigma^*$ gilt
    $$x\,z \in L \quad \Leftrightarrow \quad y\,z \in L \ .$$
  - Die Menge aller zu x äquivalenten Worte wird als Äquivalenzklasse von x bezeichnet
    $$[x]_L = \{w \in \Sigma^* \mid w \equiv_L x\}$$
  - Die Anzahl der Äquivalenzklassen wird Index bezeichnet

- **1. Beispiel: $B = \{0^n1^n \mid n \geq 0\}$**
  - Worte in B: {ε, 01, 0011, 000111,...}
  - Äquivalente Worte:
  - $1 \equiv_B 10 \equiv_B 11 \equiv_B 100 \equiv_B 101 \equiv_B$ ...
    - kein angehängtes Wort kann das Wort zu einem Wort der Sprache ergänzen
  - $01 \equiv_B 0011 \equiv_B$ ...
    - weil nur ε angehängt werden darf damit das Wort in B ist
  - $001 \equiv_B 00011 \equiv_B 0000111$
    - nur durch Anhängen von 1 kann ein Wort in B erzeugt werden

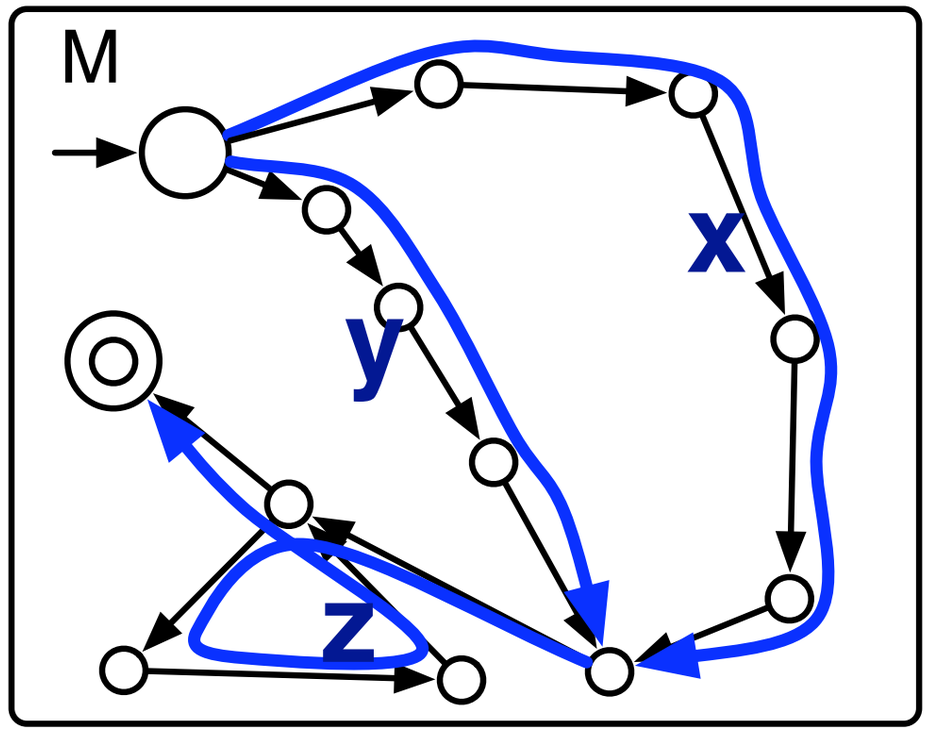
Äquivalenzklassen: Beispiel

- **1. Beispiel:**
  - Betrachte $B = \{0^n1^n \mid n \geq 0\}$
    $= \{\varepsilon, 01, 0011, 000111, ...\}$
  - Es gibt folgende Äquivalenzklassen:
    - $B_\varepsilon = \{\varepsilon\}$
    - $B_\_ = \{0^n1^m \mid m>n\geq 0\} \cup L(\Sigma^* 1 \Sigma^* 0 \Sigma^*)$
    - $B_0 = \{0^n1^n \mid n\geq 1\}$
    - $B_1 = \{0^{n+1}1^n \mid n\geq 0\}$
    - $B_2 = \{0^{n+2}1^n \mid n\geq 0\}$
    - $B_3 = \{0^{n+3}1^n \mid n\geq 0\}$
    - ...
  - Der Index von B ist unendlich.

- **2. Beispiel: Für** $\Sigma = \{0,1\}$
  - $A = L(\Sigma^* 0 \Sigma)$
    $= \{00,01,000,001,100,101,...\}$
- **Äquivalente Worte:**
  - $\varepsilon \equiv_A 1 \equiv_A 11 \equiv_A 011 \equiv_A 111 \equiv_A 0011 \equiv_A$ **...**
  - $01 \equiv_A 001 \equiv_A 101 \equiv_A 0001 \equiv_A$ **...**
  - $0 \equiv_A 10 \equiv_A 010 \equiv_A 110 \equiv_A 0010 \equiv_A$ **...**
  - $00 \equiv_A 000 \equiv_A 100 \equiv_A 0000 \equiv_A$ **...**
- **Äquivalenzklasen:**
  - $[00]_A, [01]_A, [10]_A, [11]_A$
- **Der Index von A ist 4**

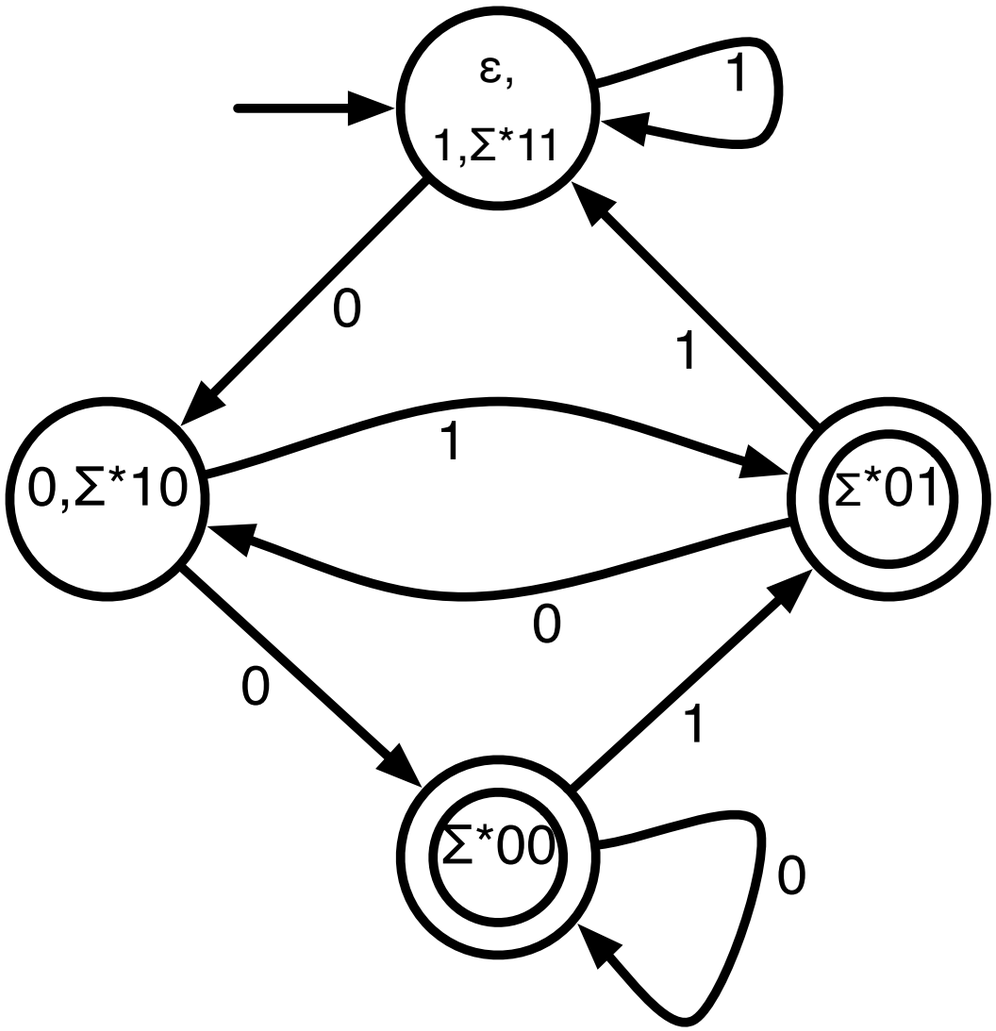
Äquivalenzklassen beschreiben das Gedächtnis eines DFA

- **Lemma:**
  - Falls $x \equiv_L y$ und $x \in L$, dann gilt $y \in L$
  - Falls $x \equiv_L y$, dann gilt $xa \equiv_L ya$ für alle $a \in \Sigma$
- **Beweis:**
  - folgt direkt aus der Definition
- **Hat ein DFA M den Zustand q mit zwei verschiedenen Worten x und y erreicht, dann gilt:**
  - $x \equiv_{L(M)} y$
  - Denn im folgenden wird M sich völlig identisch verhalten.
  - Gibt es zwei Zustände im DFA, ab der für jedes folgende Teilwort das gleiche Ergebnis herauskommt, so können sie zu einem vereinigt werden.
- **Idee: der Index beschreibt die Anzahl der Zustände des minimalen Automaten**



Der Satz von Myhill-Nerode

- **Theorem (Teil 1)**
  - Ist der Index einer Sprache A gleich k, dann gibt es einen DFA mit k Zuständen der A akzeptiert.
- **Beweis**
  - Konstruiere DFA M= (Q, Σ, δ, q_0, F) mit
    - $Q = \{[x]_A \mid x \in \Sigma^*\}$
    - $\delta([x]_A, a) = [xa]_A$
    - $q_0 = [\varepsilon]_A$
    - $F = \{[w]_A \mid w \in A\}$
  - Nach dem letztem Lemma akzeptiert M das Wort w gdw. $w \in A$
- **2. Beispiel:**
  - $A = \Sigma^* 0 \Sigma$
  - Äquivalenzklasen: $[00]_A$, $[01]_A$,$[10]_A$,$[11]_A$



Der Satz von Myhill-Nerode

- **Theorem (Teil 2)**
  - Jeder DFA M mit k Zuständen akzeptiert eine Sprache mit Index ≤ k.
- **Betrachte einen Zustand q des DFA**
  - Definiere: $L_q = \{w \in \Sigma^* \mid \delta(q_0,w) = q\}$
    - wobei $\delta(q,wa) := \delta(\delta(q,w),a)$ für $a \in \Sigma$
  - Beweis:
    - Für alle $x,y \in Lq$ und $z \in \Sigma^*$ gilt:
      - $\delta(q,z) = \delta(q_0,xz) = \delta(q_0,yz)$
    - Also $x \equiv_{L(M)} y$
- **Wenn jeder der k Zustände nur äquivalente Worte hat, dann kann es höchstens k Äquivalenzklassen geben**
- **Behauptung: L_q beinhaltet nur äquivalente Worte bezüglich L = L(M)**

Der minimale endliche deterministische Automat

- **Korollar**
  - Die Anzahl der Zustände eines minimalen endlichen Automats entspricht der Anzahl der Äquivalenzklassen der zugehörigen Sprache
- **Beweis:**
  - Es gibt einen DFA mit k=Index Zuständen
  - Jeder DFA hat mindestens k Zustände
  - Daher ist der durch die Äquivalenzklassen definierte Automat minimal.
- **Mit der Kenntniss der Äquivalenzklassen lässt sich ein minimaler DFA konstruieren**

Äquivalenzklassen zum Beweis der Nichtregularität

- **Theorem (Teil 2)**
  - Jeder DFA M mit k Zuständen akzeptiert eine Sprache mit Index ≤ k.
- **Beweis**
  - Ist der Index einer Sprache unendlich, dann kann es keinen endlichen Automaten geben, der die Sprache akzeptiert.
- **Aus der Kenntnis unendlicher Äquivalenzklassen lässt sich also die Nichtregularität beweisen**

Beispiel

- **Betrachte B = {0^n1^n | n ≥ 0} = {ε,01,0011,000111,...}**
  - Es gibt folgende Äquivalenzklassen:
    - $B_\varepsilon = \{\varepsilon\}$
    - $B\_ = \{0^n1^m \mid m>n\geq 0\} \cup L(\Sigma^*1\ \Sigma^*\ 0\ \Sigma^*)$
    - $B_0 = \{0^n1^n \mid n\geq 1\}$
    - $B_1 = \{0^{n+1}1^n \mid n\geq 0\}$
    - $B_2 = \{0^{n+2}1^n \mid n\geq 0\}$
    - $B_3 = \{0^{n+3}1^n \mid n\geq 0\}$
    - ...
  - Der Index von B ist unendlich.
- **Also ist B nach dem Satz von Myhill-Nerode nicht regulär.**

Formale Sprachen und
Endliche Automaten

Ende