# *Informatik III*

**Christian Schindelhauer**
Wintersemester 2006/07
4. Vorlesung
03.11.2006
**schindel@informatik.uni-freiburg.de**

**Albert-Ludwigs-Universität Freiburg**
**Rechnernetze und Telematik**
**Prof. Dr. Christian Schindelhauer**

# Reguläre Operationen

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

1. **Vereinigung**:

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

2. **Konkatenation**

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

3. **Stern**

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

# Die regulären Ausdrücke beschreiben genau die reguläre Sprachen (1. Teil: ⇒)

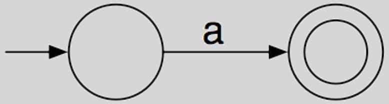
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Lemma**
  - Jeder reguläre Ausdruck R beschreibt eine reguläre Sprache

**Beweis**
  - Wir konvertieren R in einen NFA

**1. Fall R = a, für a ∈ Σ**

Automat:

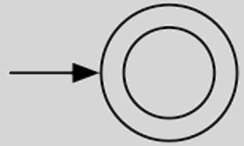



– Formal:

$$N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$$
$$\delta(q_1, a) = \{q_2\}$$
$$\delta(r, b) = \emptyset, \text{ für } r \neq q_1 \text{ oder } b \neq a$$

**2. Fall R = ε**

Automat:

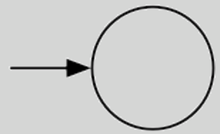


– Formal:

$$N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$$
$$\delta(r, b) = \emptyset, \text{ für alle } r,\ b$$

**3. Fall R = ∅**

Automat:



– Formal:

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

**4. Fall: R = ($R_1 \cup R_2$),**

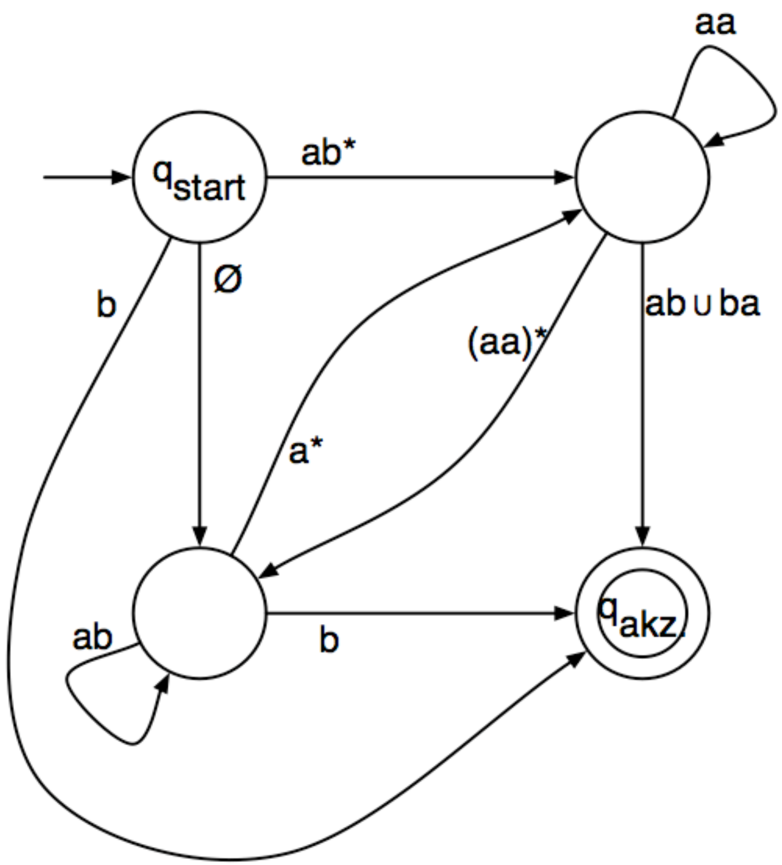
**5. Fall: ($R_1 \circ R_2$),**

**6. Fall: ($R_1$)***

**siehe Folien 03-3 bis 03-08**

# Die regulären Ausdrücke beschreiben genau die reguläre Sprachen (2. Teil: $\Leftarrow$)

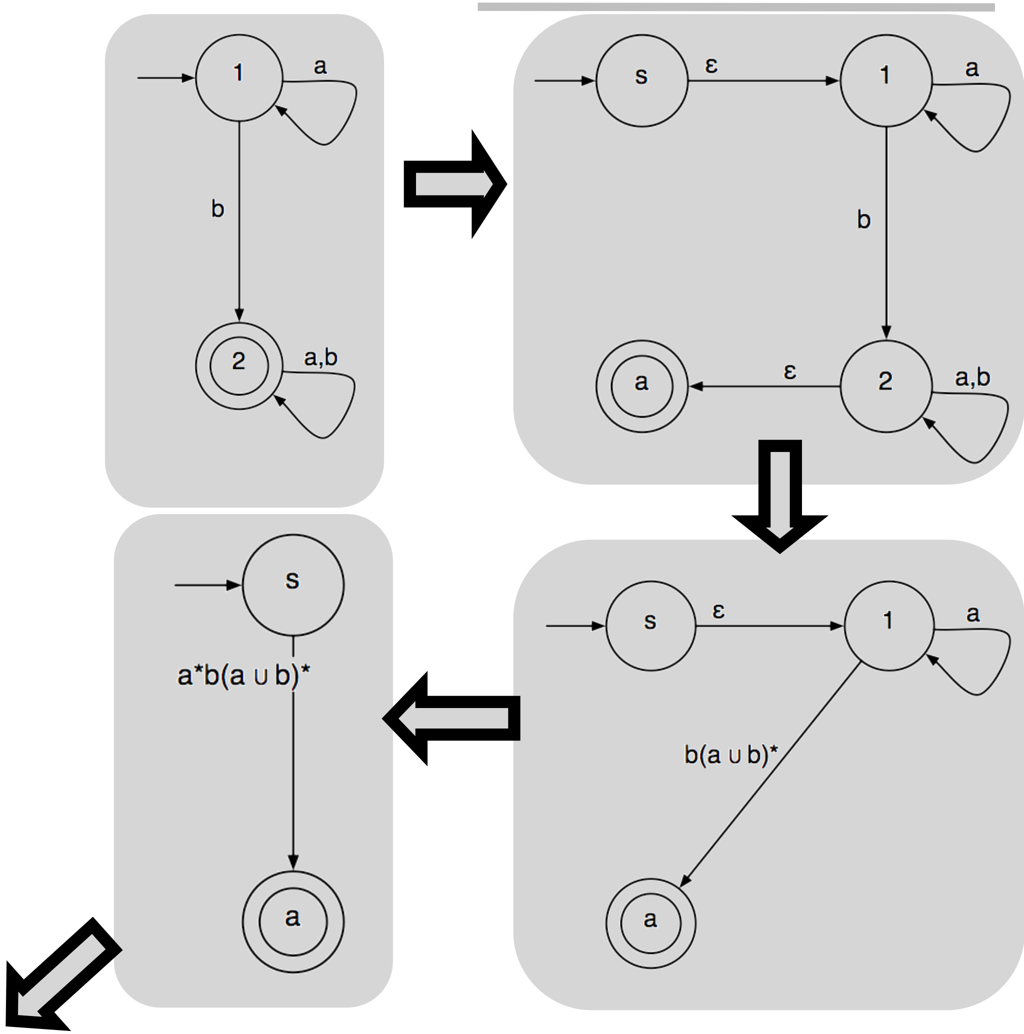
- **Strategie:**
  - Einführung der **verallgemeinerten nichtdeterministischen endlichen Automaten** (**G**eneralized **N**on-deterministic **F**inite **A**utomata - GNFA)
  - NFA → GNFA
  - GNFA → Regulärer Ausdruck
- **Eigenschaften GNFA:**
  - GNFA = NFA + reguläre Ausdrücke
  - Reguläre Ausdrücke auf den Übergängen
  - Ein akzeptierender Zustand
  - Alle Übergänge existieren
  - Ausnahmen:
    - Kein Übergang hin zum Startzustand
    - Kein Übergang ausgehend vom akzeptierenden Zustand



# Die regulären Ausdrücke beschreiben genau die reguläre Sprachen (2. Teil: ⇐)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

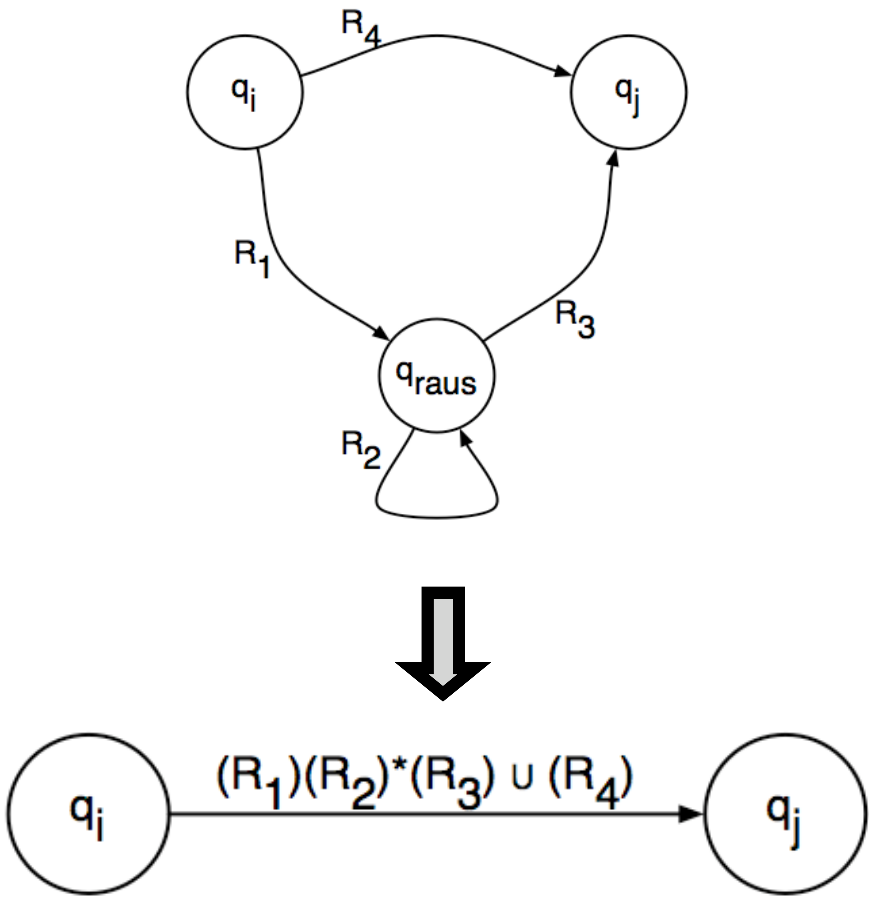
- **Strategie:**
  - NFA mit k Zuständen
    → GNFA mit k+2 Zuständen
  - GNFA mit k+2 Zuständen
    → GNFA mit k+1 Zuständen
  - GNFA mit k+1 Zuständen
    → GNFA mit k Zuständen
  - ...
  - GNFA mit 3 Zuständen
    → GNFA mit 2 Zuständen
  - GNFA mit 2 Zuständen
    → Regulärer Ausdruck



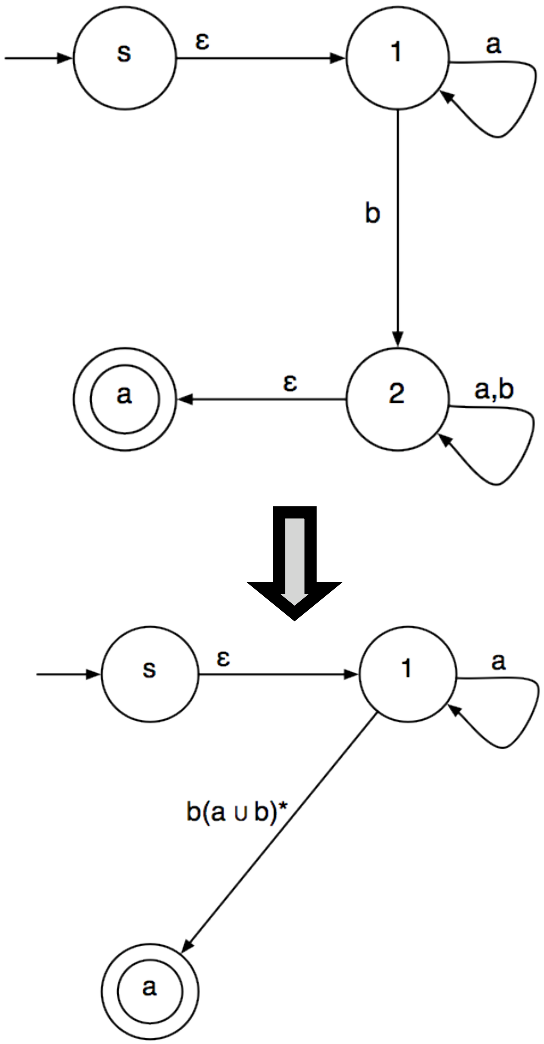

`a*b(a ∪ b)*`

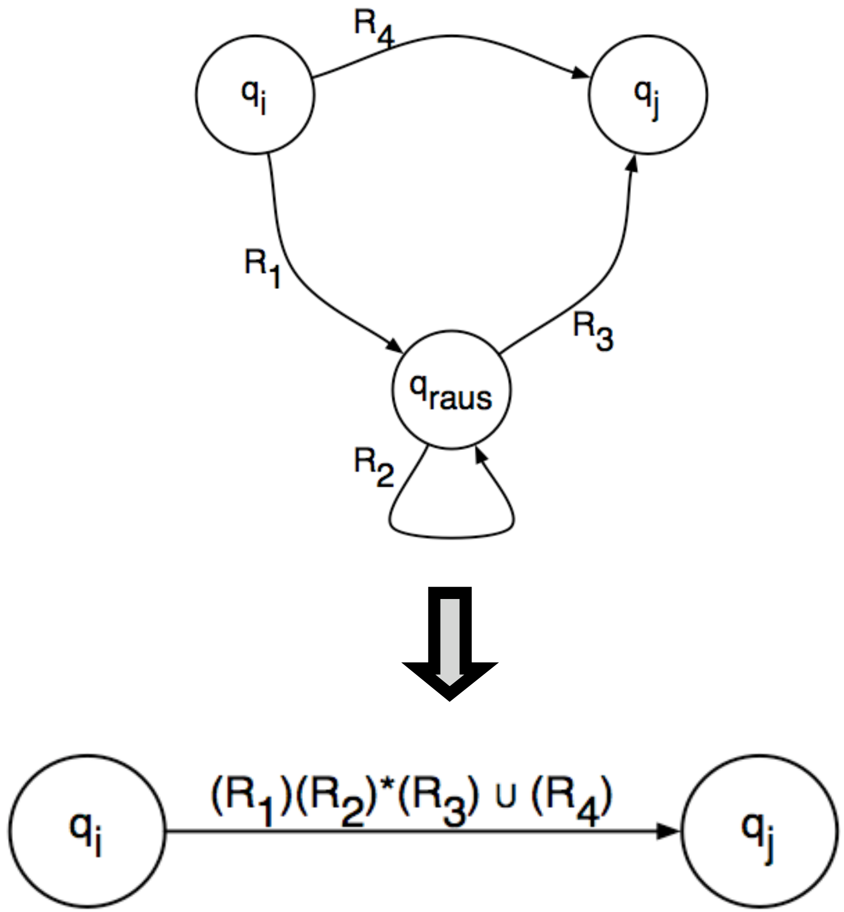
# GNFA mit k Zuständen → GNFA mit k-1 Zuständen

- **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 ($R_4$)



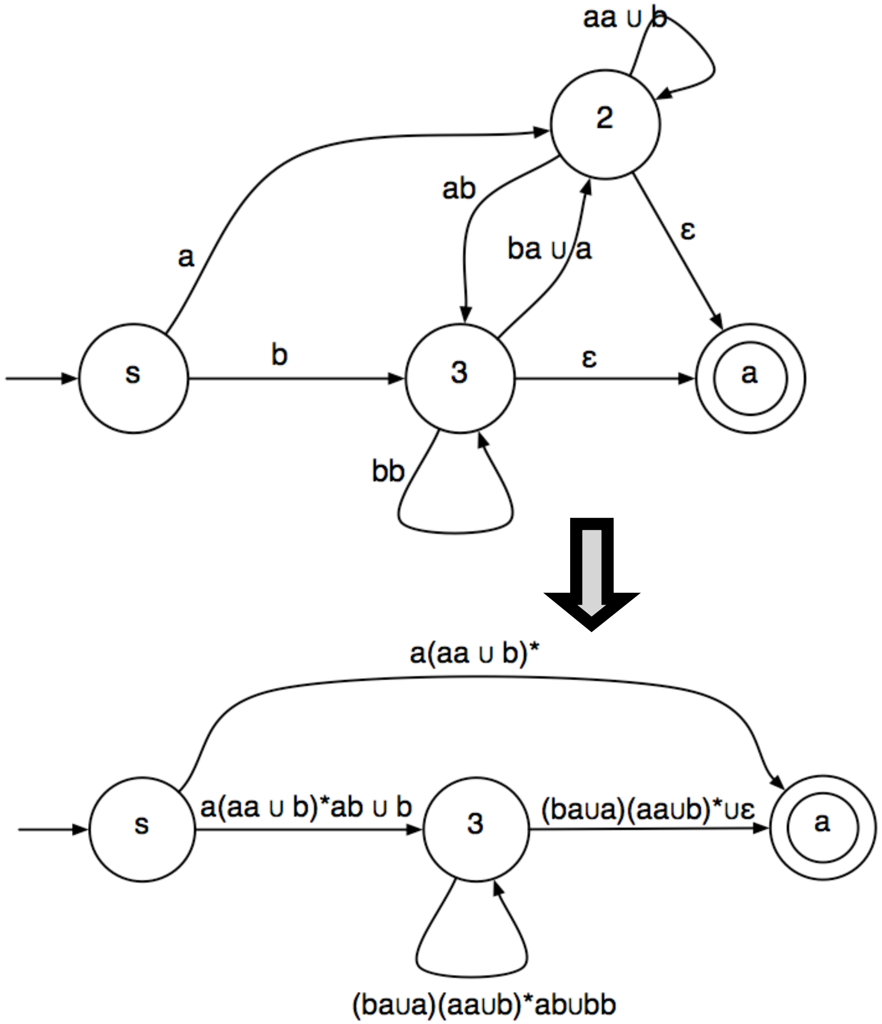
# GNFA mit k Zuständen → GNFA mit k-1 Zuständen

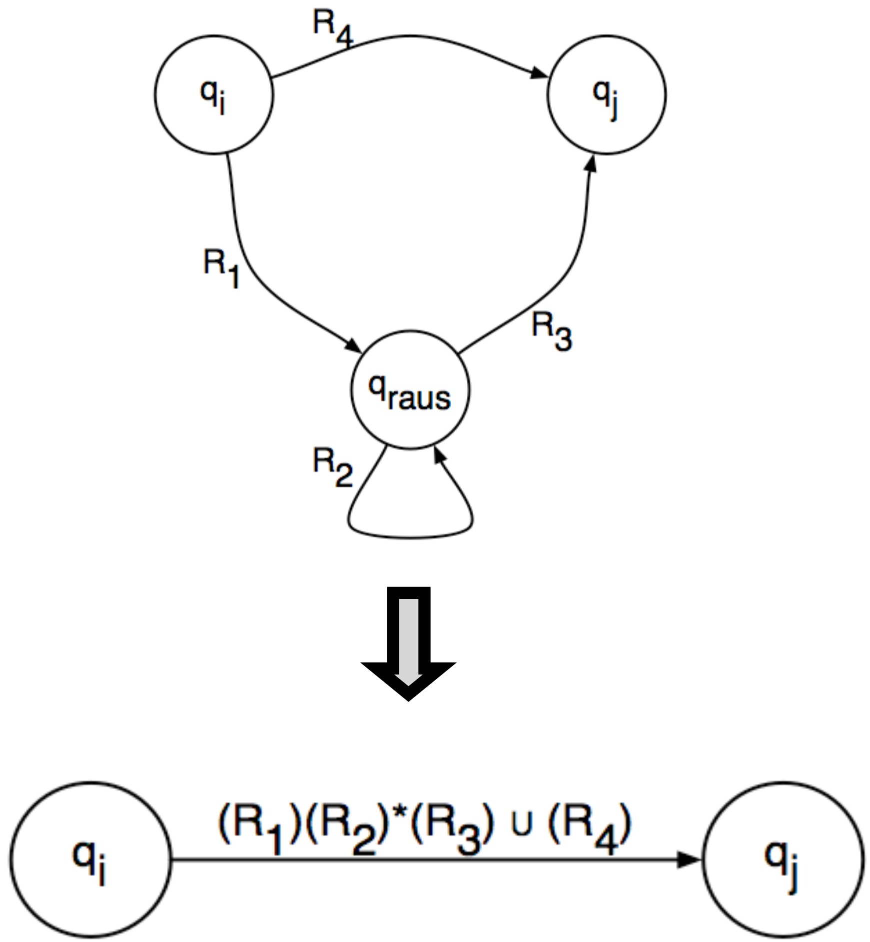


Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# GNFA mit k Zuständen → GNFA mit k-1 Zuständen



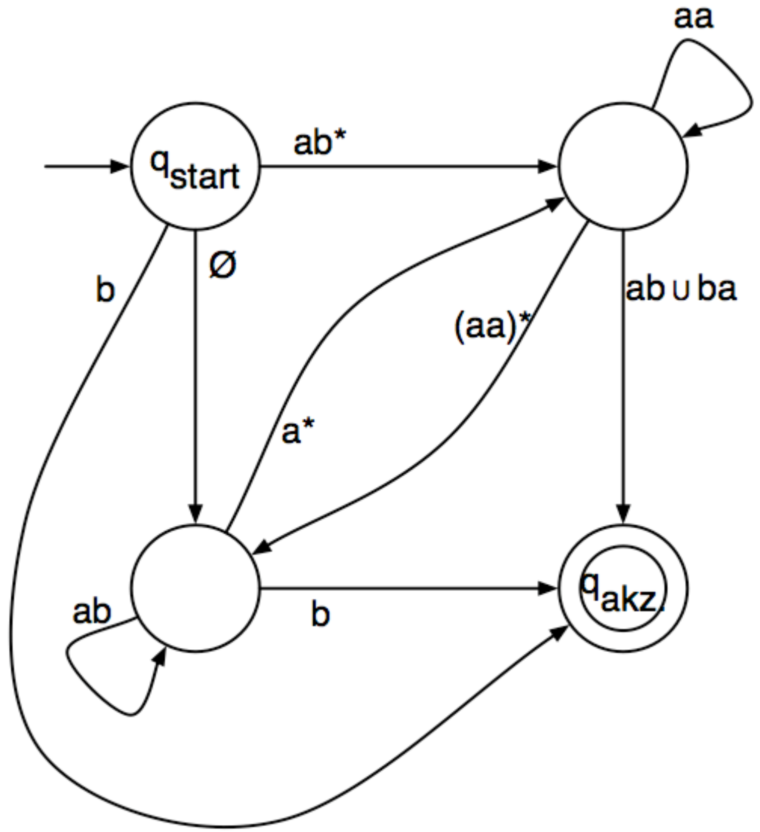



# Und jetzt ausführlich:

- **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
    1. Q: Eine endliche Menge von **Zuständen**
    2. $\Sigma$: ist das **Alphabet**
    3. $\delta: (Q \setminus \{q_{akz}\}) \times (Q \setminus \{q_{start}\}) \rightarrow$ **R** ist die **Übergangsfunktion**
       - R ist die Menge der regulären Ausdrücke
    4. $q_{start} \in Q$: ist der **Startzustand**
    5. $q_{akz} \in Q$: ist der **akzeptierende Endzustand**

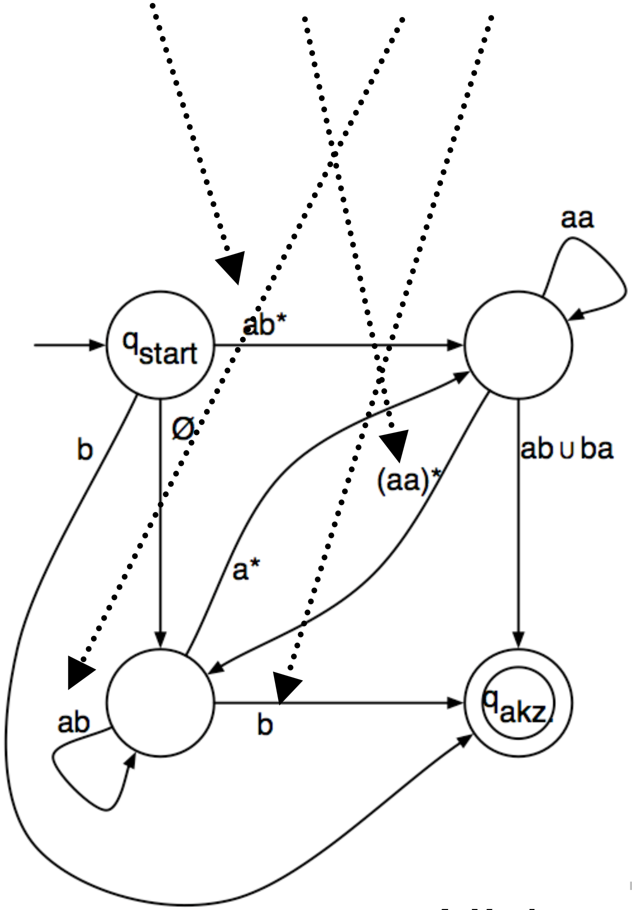


# Berechnung eines GNFA

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

- **w = abbbaaaaabb**

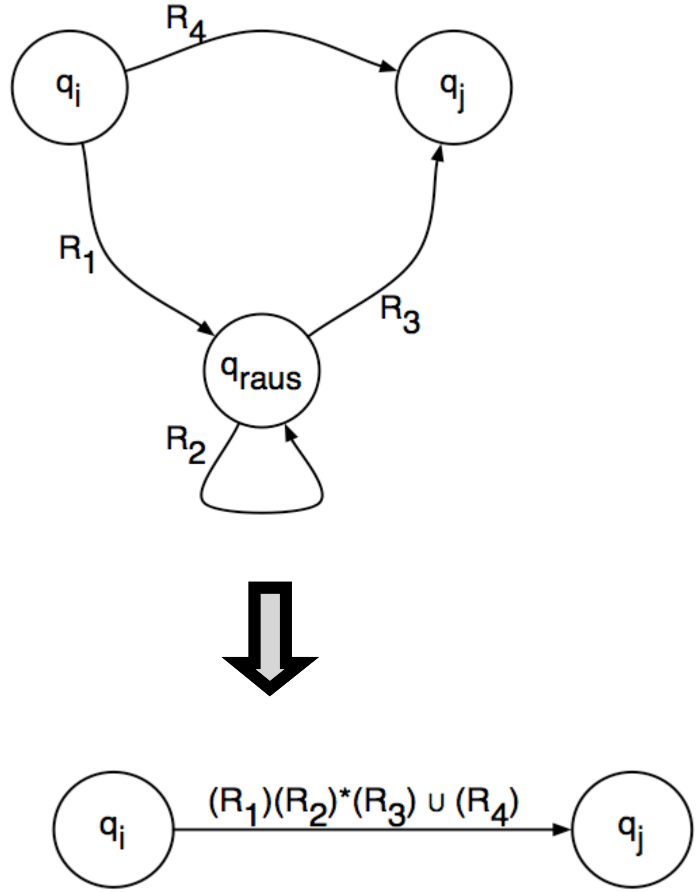
  **= abbb aaaa ab b**



# Der KONVERTIERER

**Konvertiere(G=(Q, $\Sigma$, $\delta$, $q_{start}$, $q_{akz}$) :GNFA):**

1. **Sei k die Anzahl der Zustände von G**
2. **Falls k=2 dann**
   - Gib regulären Ausdruck zwischen Startzustand und akzept. Zustand aus
3. **Falls k>2 dann**
   - Wähle beliebig $q_{raus} \in Q \backslash \{q_{akz}, q_{start}\}$ aus
   - Betrachte GNFA
     $G' = (Q', \Sigma, \delta', q_{start}, q_{akz})$ mit
     - $Q' = Q \setminus \{q_{raus}\}$
     - Für alle $q_i \in Q \backslash \{q_{akz}\}$ und $q_j \in Q \backslash \{q_{start}\})$ 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') und gib das Ergebnis aus

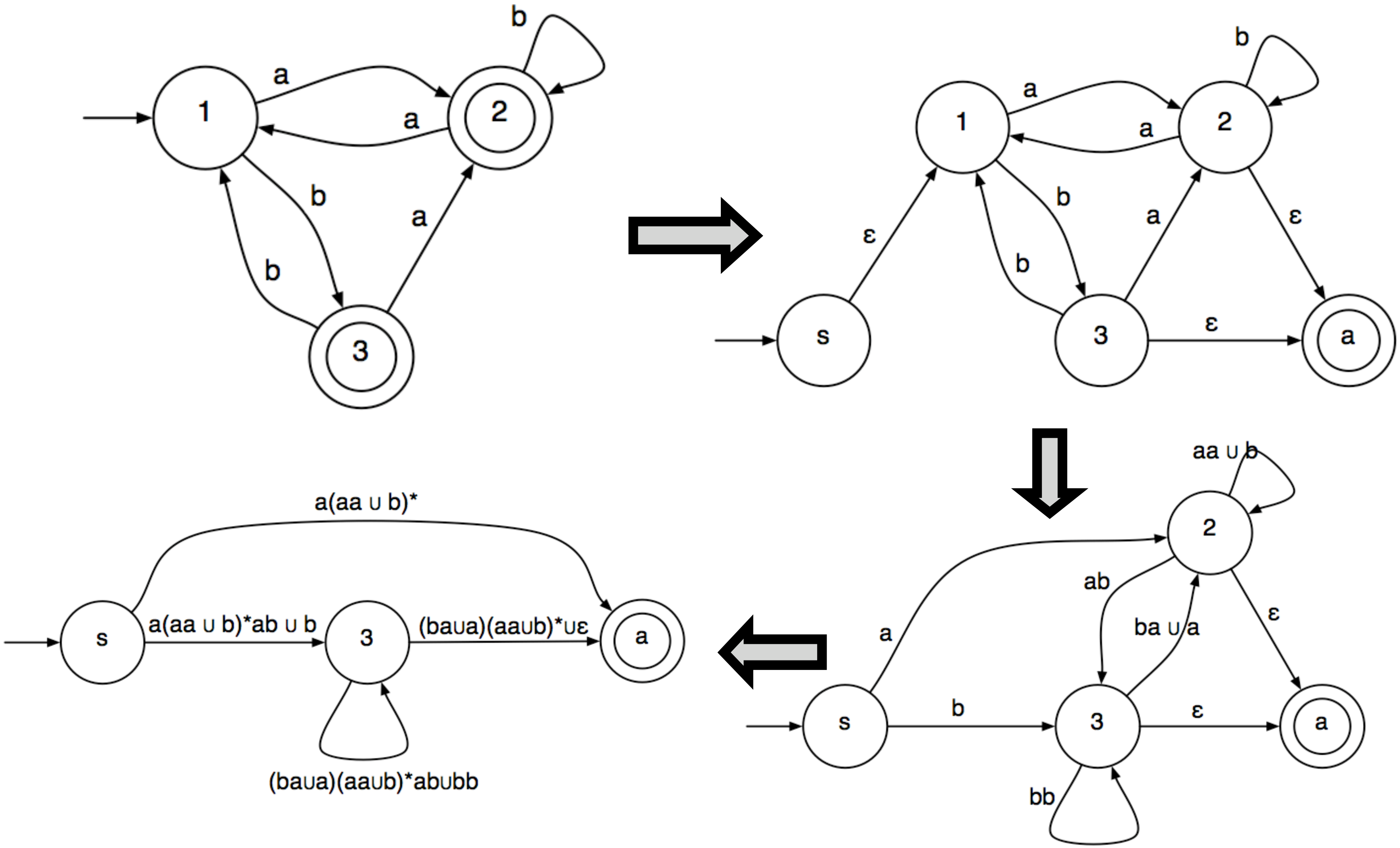


# Beweis der Korrektheit

- **Behauptung**
  - Für jeden GNFA G ist Konvertiere(G) äquivalent zu G.
- **Beweis durch Induktion über die Anzahl der Zustände von G k**

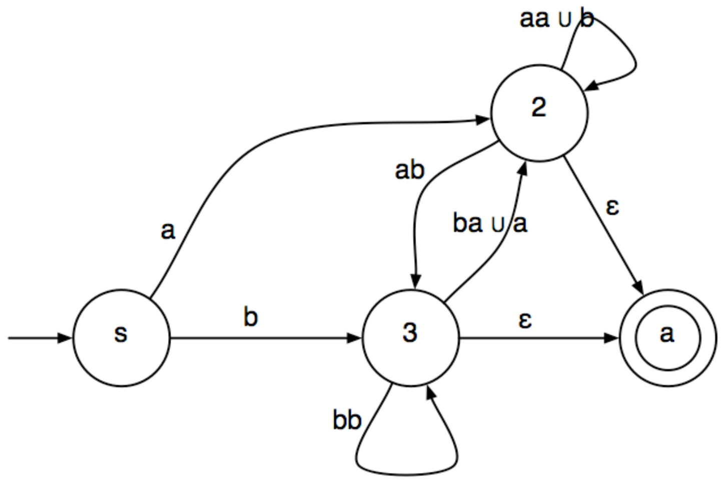
1. **Basis (Anker): k=2**
   - Der GNFA G hat nur einen Übergang
   - Nach Definition ist er äquivalent mit dem regulären Ausdruck auf dem Übergang
2. **Induktionsschritt: Angenommen die Behaupting ist wahr für k-1 Zustände**
   - Angenommen G akzeptiert w und sei $q_{start}$, $q_1$, $q_2$, ... $q_{akz}$ 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 und 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 $w' \in L((R_1)\ (R_2)^*\ (R_3))$
     - In jedem Fall würde auch G das Wort w akzeptieren
   - **G akzeptiert also gdw. G' akzeptiert**

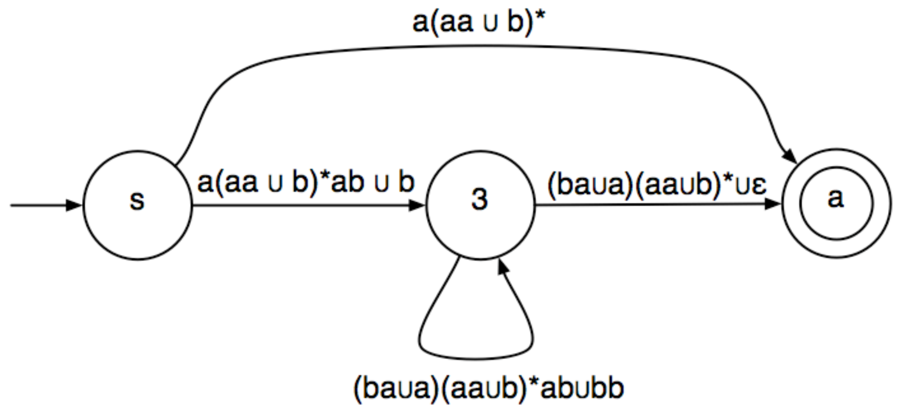
# Noch ein Beispiel



Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Noch ein Beispiel




(a(aa ∪ b)*ab ∪ b) ((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)*∪ε) ∪ a(aa ∪ b)*

# Daraus folgt...

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

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Pumping-Lemma und Minimale Automaten

# Das Pumping-Lemma

- **Motivation**
  - Reguläre Sprachen sind die Stottottottotterer der Sprachen
  - Was ist nicht regulär?
    - Palindrome,

$$\text{Palindrom} = \{w \in \{a,b\}^* \mid \forall i \in \{1,\ldots,|w|\} \; : \; w_i = w_{|w|-i+1}\}$$
$$= \{\epsilon, a, b, aa, bb, aaa, aba, bab, bbb, \ldots\}$$

    - Kopie = { w w | w ∈ Σ*}
    - Klammersprachen, z.B. Syntaktische Überprüfung von
      - `((a+b)-b = (a))/(b))+b-a`
    - Zählsprachen, z.B.
      - { w ∈ {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**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Das Pumping-Lemma

## ➢ Pumping-Lemma

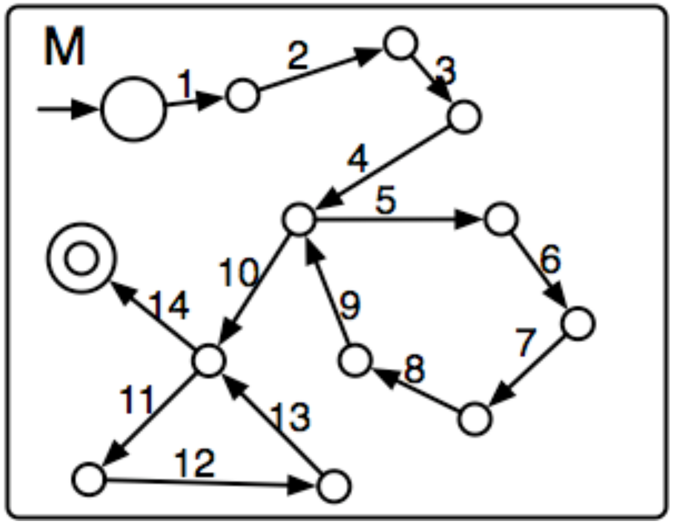
– Sei A eine reguläre Sprache.
- Dann gibt es eine Zahl $p>0$
- so dass für jedes Wort s mit $|s| \geq p$
- s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
  - für alle $i \geq 0$: $xy^iz \in A$
  - $|y| > 0$
  - $|xy| \leq p$.

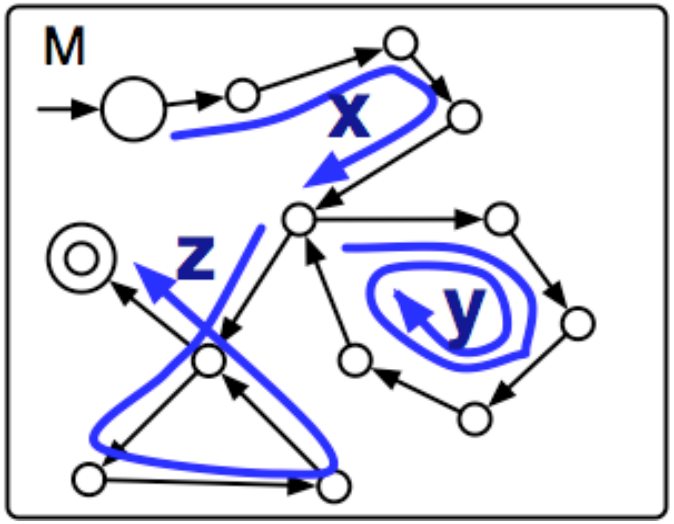
## ➢ Was heißt das:

– Betrachte Sprache:
- A = L(**Stott**(**ott**)*(**er**)*)
- p = 8
- s = **Stottotterererer**
  - x=**Stott**
  - y=**ott**
  - z=**erererer**
- $y \neq \varepsilon$
- $|xy| \leq 8$
- All diese Worte sind in A:
  - **Stotterererer**
  - **Stottotterererer**
  - **Stottottotterererer**
  - **Stottottottotterererer**
  - **Stottottottottotterererer**
  - **Stottottottottottotterererer**
  - **Stottottottottottottotterererer**
  - ...................................

# Pumping-Lemma
# Beweisidee

- **Pumping-Lemma**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl p>0
    - so dass für jedes Wort s mit $|s| \geq p$
    - s in drei Teile geteilt werden kann: s = xyz, wobei gilt
      - für alle $i \geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.
- **Beweisidee**
  - Es gibt nur endlich viele Zustände im DFA von A
  - Auf langen Worten wiederholen sich manche Zustände
  - Das Wort dazwischen (nämlich y) kann also beliebig oft eingesetzt werden!



# Der Beweis des Pumping-Lemmas

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Pumping-Lemma**
  - Sei A eine reguläre Sprache.
    - Dann gibt es eine Zahl $p>0$
    - so dass für jedes Wort s mit $|s|\geq p$
    - s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
      - für alle $i\geq 0$: $xy^iz \in A$
      - $|y| > 0$
      - $|xy| \leq p$.
- **Beweis:**
  - Sei $M= (Q, \Sigma, \delta, q_0, F)$ ein DFA der A akzeptiert mit $p=|Q|$ Zuständen
  - Sei $s = s_1s_2...s_n$ ein Wort in A der Länge $n\geq p$
  - Sei $q = r_1r_2... r_{n+1}$ die Folge der Zustände in M bei der Berechnung von s.
  - Diese Folge hat Länge $n+1 \geq p+1$
  - 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 $k \leq p+1$
  - 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.
- **Damit folgt das Pumping-Lemma**

# Das Pumping-Lemma ist ein Killerargument

- **Beispiel:**
  - Sei $B = \{0^n 1^n \mid n \geq 0\}$
- **zu zeigen: B ist nicht regulär**
- **Angenommen doch**
- **(Pumping-Lemma)**
  - Dann gibt es eine Zahl p>0
  - so dass für jedes Wort s 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 $xyyz = 0^{p+m}1^p$
  - i=3: Das Wort $xy^3z = 0^{p+2m}1^p$
  - ...
- **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**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

# Das Pumping-Lemma noch einmal

- **Betrachte die Sprache**
  - $F = \{ww \mid w \in \{0,1\}^*\}$
- **zu zeigen: F ist nicht regulär**
- **Angenommen doch**
- **(Pumping-Lemma)**
  - Dann gibt es eine Zahl $p>0$
  - so dass für jedes Wort s mit $|s| \geq p$
  - s in drei Teile geteilt werden kann: $s = xyz$, wobei gilt
    - für alle $i \geq 0$: $xy^iz \in F$
    - $|y| > 0$
    - $|xy| \leq p$.
- **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

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

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

# *Ende der 4. Vorlesung*

CoNe Freiburg

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Informatik III

**Christian Schindelhauer**
**schindel@informatik.uni-freiburg.de**