Theoretische Informatik

Deterministische Automaten

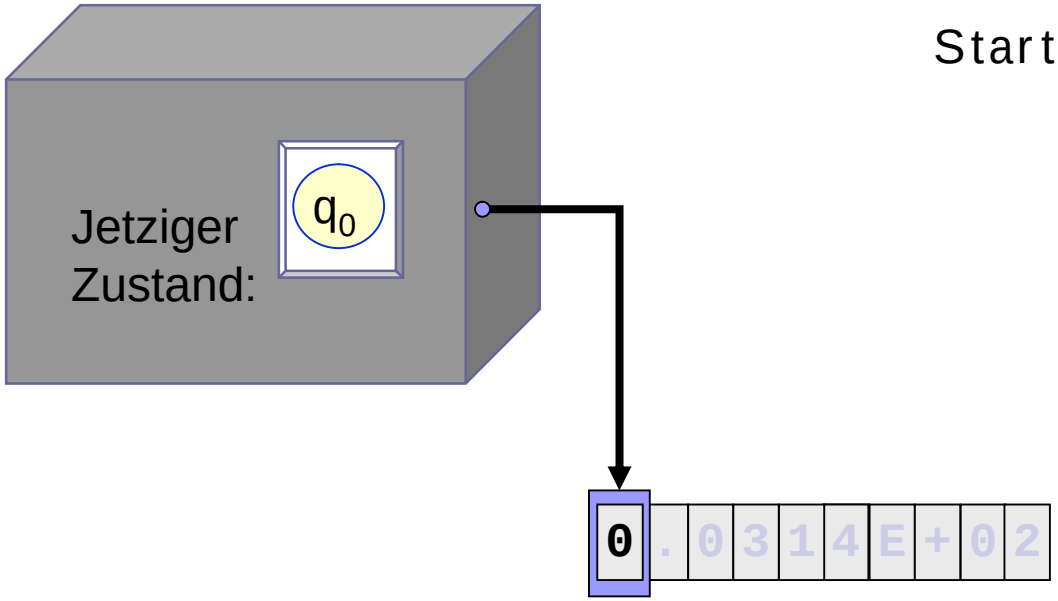
Inhalt

1. Deterministische Akzeptoren
   - Grundbegriffe
   - Sprache eines Automaten
   - Implementierung
   - Komplement, Produkte
   - Homomorphismen von Automaten
   - Faktorautomat
   - Homomorphiesatz

2. Minimierung und Grenzen von Automaten
   - Trennbarkeit
   - Nerode-Lemma
   - Pumping Lemma
   - nicht reguläre Sprachen
   - Minimierung

Automaten

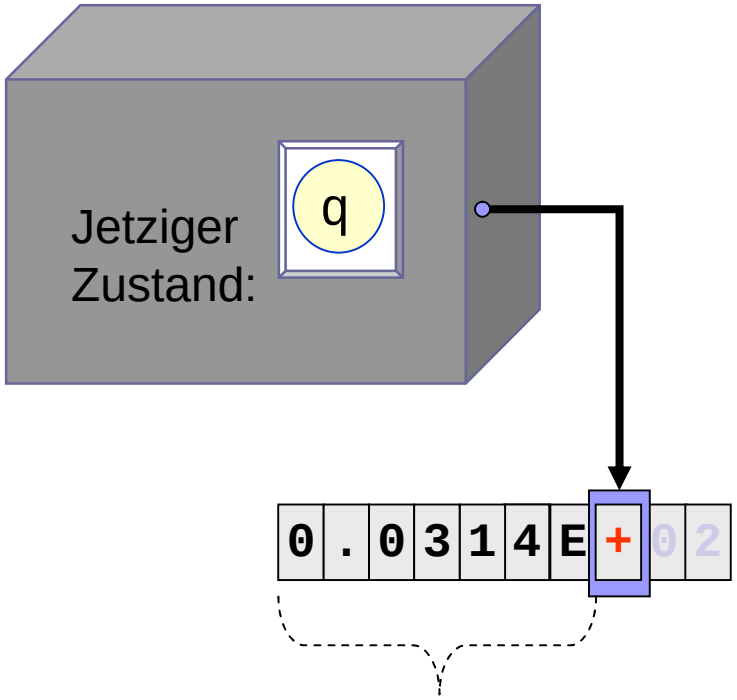
n sollen Sprache erkennen



Starten in Anfangszustand q_0

Automaten

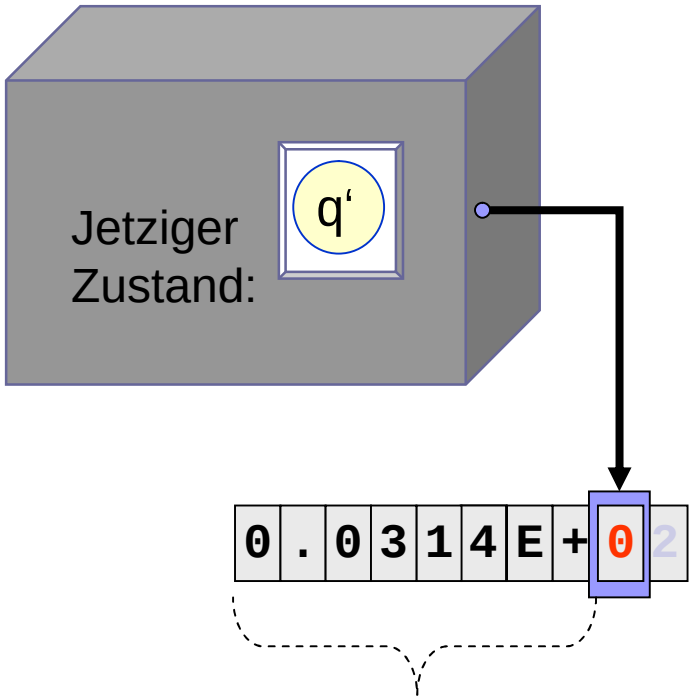
n sollen Sprache erkennen



Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

neuer Zustand $q' = \delta(q,a)$



Automaten

n sollen Sprache erkennen

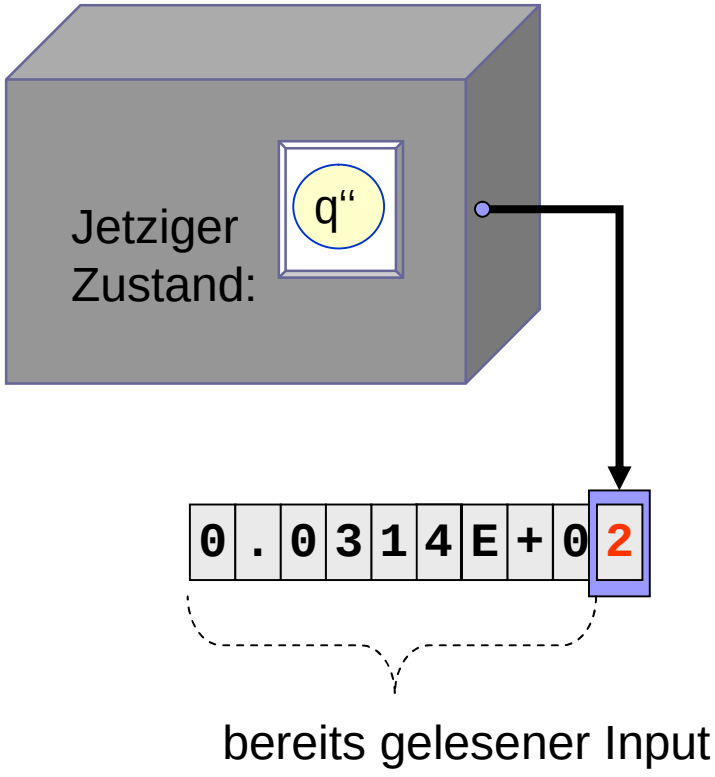
Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

neuer Zustand $q' = \delta(q,a)$

Automaten

n sollen Sprache erkennen



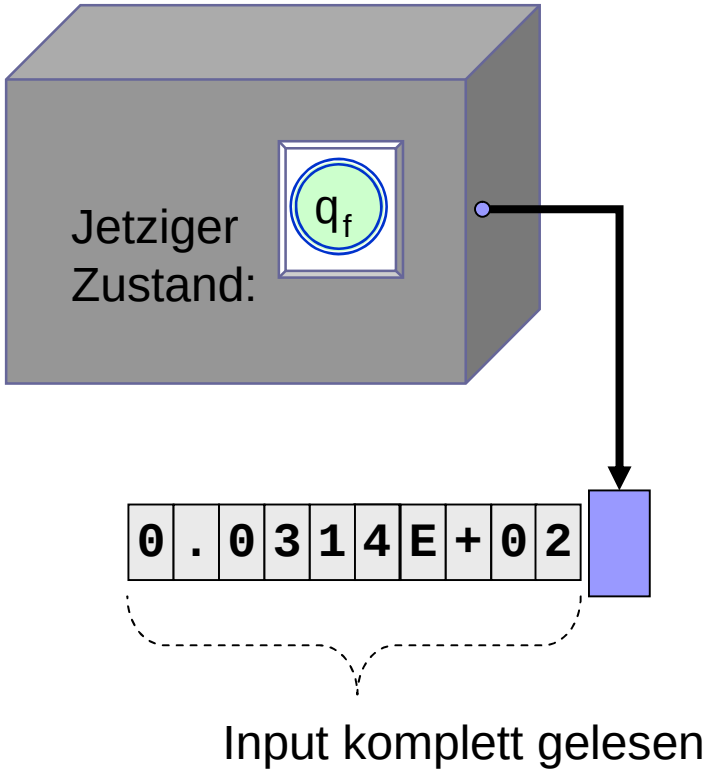
Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

neuer Zustand $q' = \delta(q,a)$

Automaten

n sollen Sprache erkennen



Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

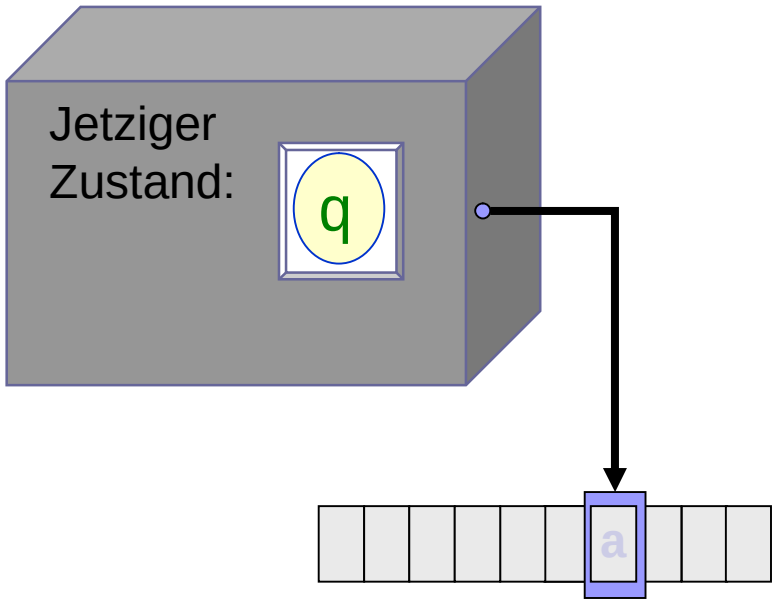
neuer Zustand $q' = \delta(q,a)$

Wenn Wort w komplett eingelesen, wird es akzeptiert, falls Automat in einem Endzustand $q_f \in F$ ist.

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q - einer endlichen Menge von Zuständen



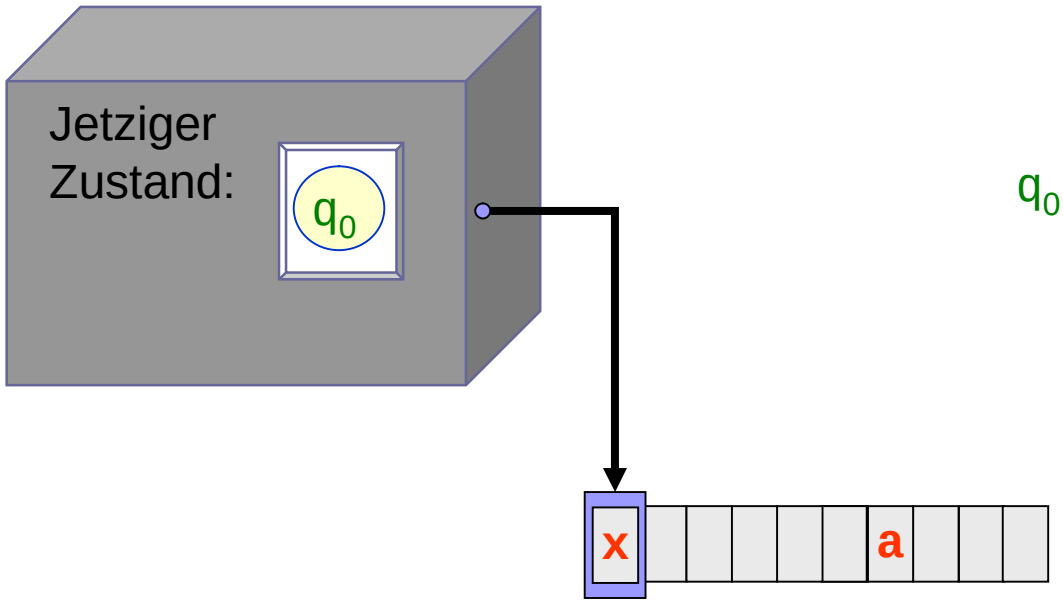
$q \in Q$

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q - einer endlichen Menge von Zuständen

$q_0 \in Q$ - einem Anfangszustand

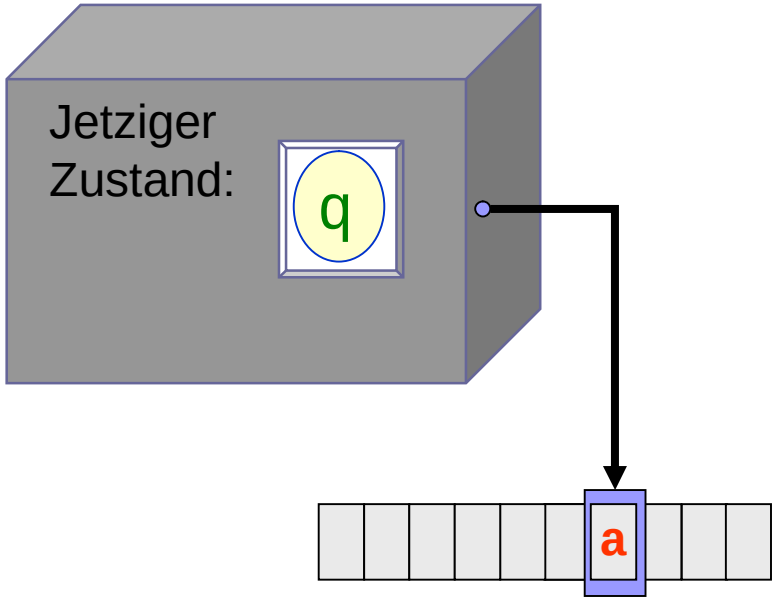


$q_0 \in Q$

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q	- einer endlichen Menge von Zuständen
$\delta : Q \times \Sigma \rightarrow Q$	- Transitionsfunktion
$q_0 \in Q$	- einem Anfangszustand



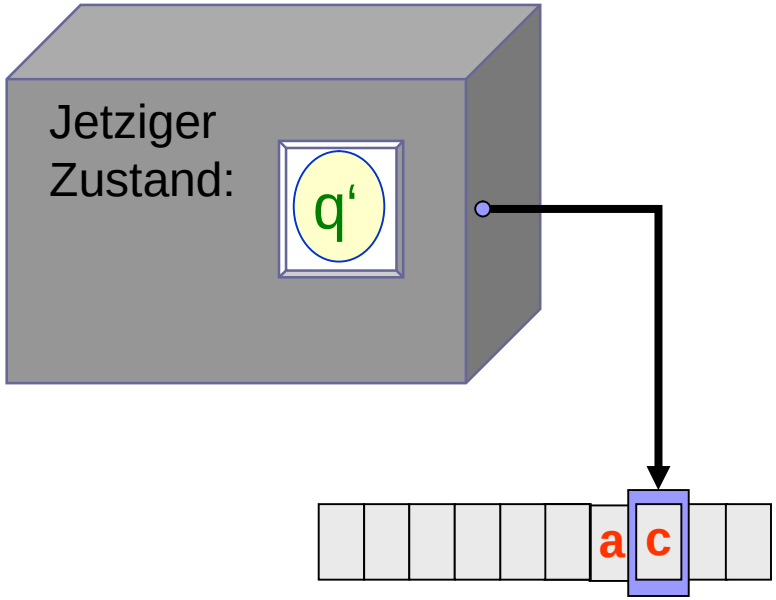
neuer Zustand

$\delta(q,a)=q'$

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q	- einer endlichen Menge von Zuständen
$\delta : Q \times \Sigma \rightarrow Q$	- Transitionsfunktion
$q_0 \in Q$	- einem Anfangszustand



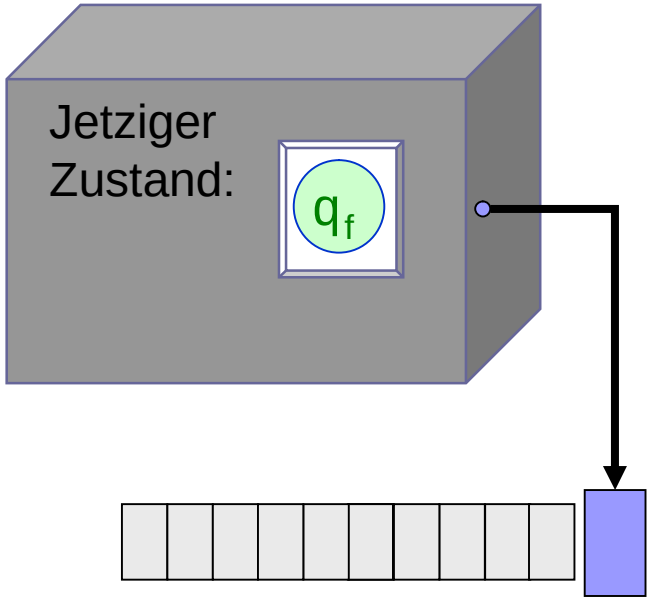
neuer Zustand

$\delta(q,a)=q'$

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q	- einer endlichen Menge von Zuständen
$\delta : Q \times \Sigma \rightarrow Q$	- Transitionsfunktion
$q_0 \in Q$	- einem Anfangszustand
$F \subseteq Q$	- einer Menge von Endzustände



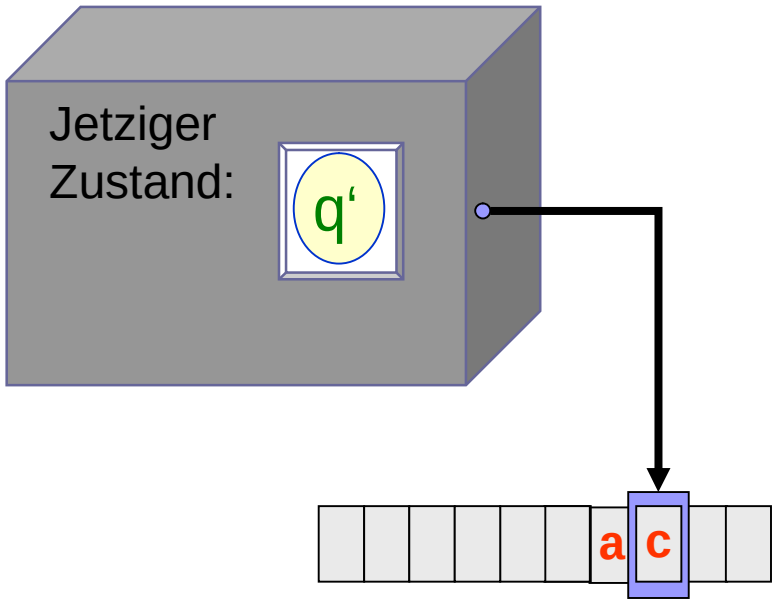
$q_f \in F \subseteq Q$

Definition: Automat (DFA)

Sei Σ ein Alphabet. Ein deterministischer endlicher Σ-*Automat A* besteht aus

Q	- einer endlichen Menge von Zuständen
$\delta : Q \times \Sigma \rightarrow Q$	- Transitionsfunktion
$q_0 \in Q$	- einem Anfangszustand
$F \subseteq Q$	- einer Menge von Endzustände

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

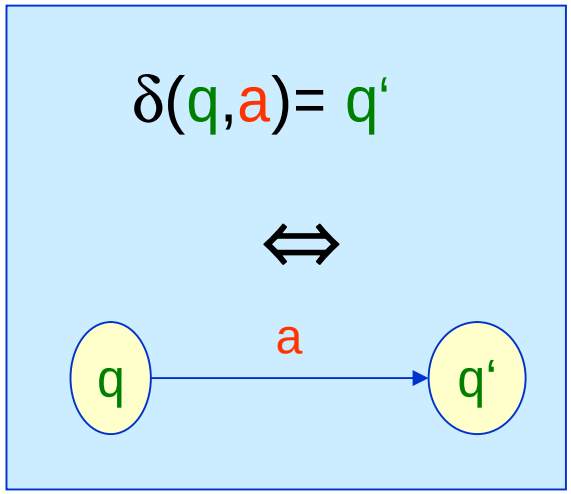


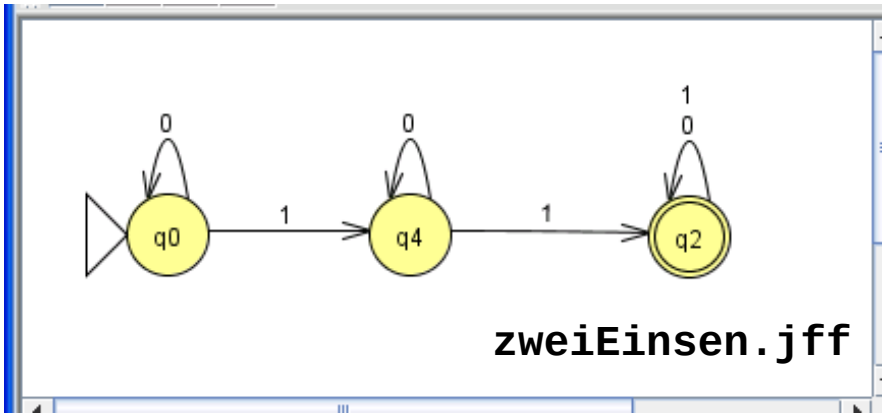
neuer Zustand

$\delta(q,a)=q'$

Zustandsübergangsgraphen

- Graphische Notation für Automaten
  - Zustände - Knoten
  - ein Anfangszustand $q_0 \in Q$
  - Zustandsübergänge – beschriftete Kanten
  - eine Menge von Endzuständen $F \subseteq Q$
- Wort wird akzeptiert, falls es
  - beginnend im Anfangszustand
  - den Automaten in Endzustand überführt





Hier z.B.:

$Q = \{q_0, q_4, q_2\}$

$F = \{q_2\}$

$\delta(q_0,0) = q_0$

$\delta(q_0,1) = q4$

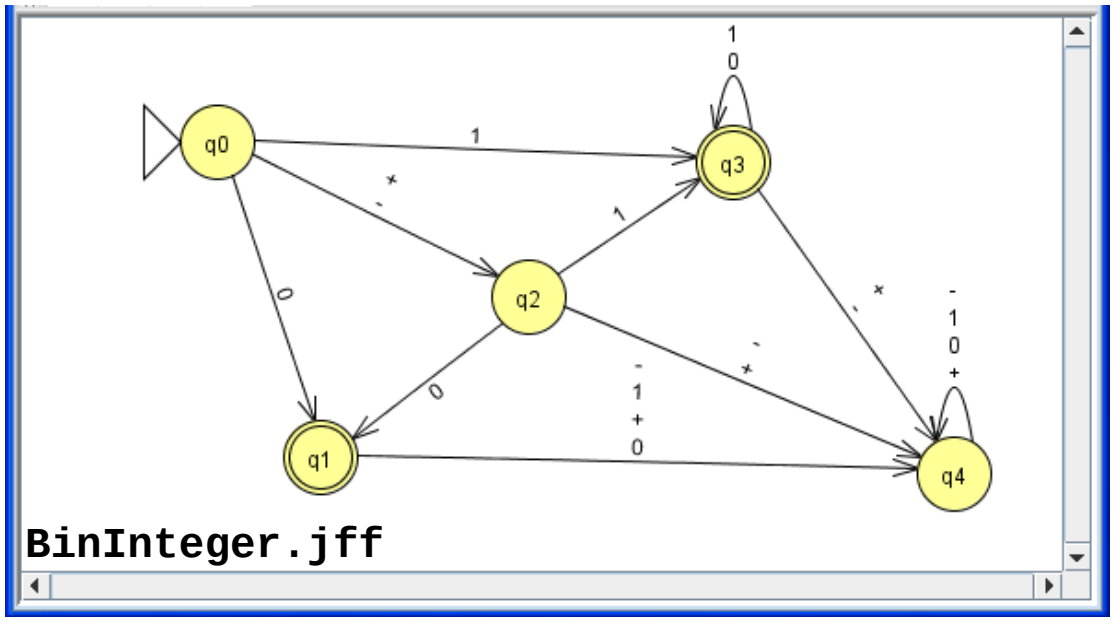
BinInteger-Automat

- Automat über dem Alphabet $\Sigma=\{0,1,+,-\}$.
- Erkennt alle gültigen Binären Integer
  - optionales Vorzeichen
  - keine führenden 0-en erlaubt
- Entspricht dem regulären Ausdruck
  - [+,-]? (0 | 1(0|1)*)

Automat:

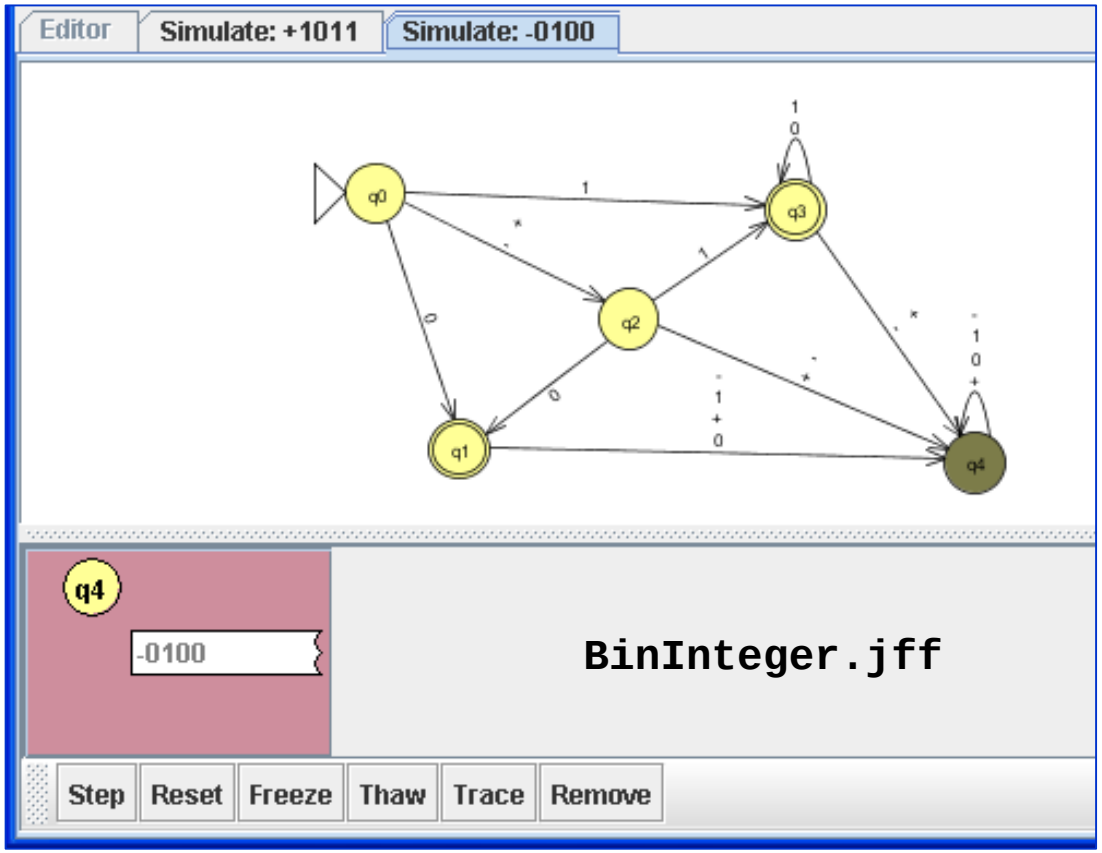
$Q=\{ q_0, q_1, q_2, q_3, q_4 \}$
q_0 ist Anfangszustand
$F=\{ q_1, q_3 \}$ Endzustände

q_4 „*Error“-Zustand*
- nicht akzeptierend
- kann nicht verlassen werden

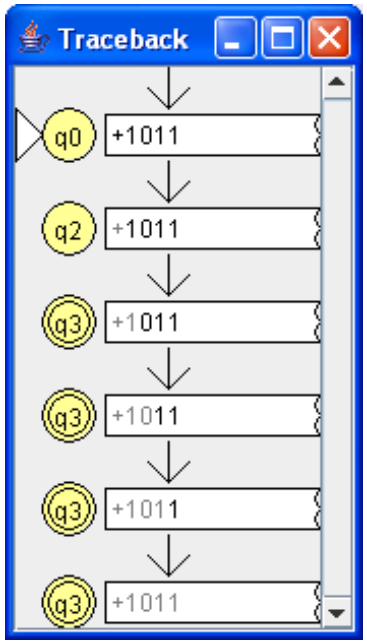


Akzeptanz

- Ein Wort $w \in \Sigma^*$ wird akzeptiert, falls es vom Anfangszustand in einen Endzustand führt

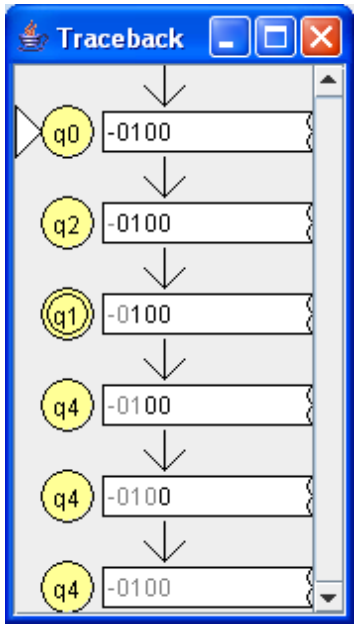


JFLAP output



+1011 führt zu $q_3 \in F$

$\Rightarrow$ akzeptiert

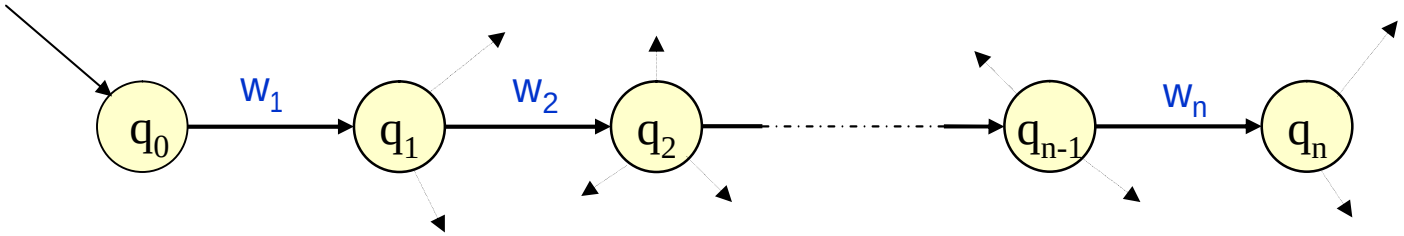


-0100 führt zu $q_4 \notin F$

$\Rightarrow$ nicht akzeptiert

Läufe

- Wort beschreibt Weg durch den Automaten
  - Wort $w=w_1w_2...w_n$ ist Fahrplan
  - Jedes Wort ist gültiger Fahrplan, weil $\delta(q,a)$ für alle q und alle a definiert
- Ein Lauf ist die Folge der dabei besuchten Zustände
  - w_i 2 Σ



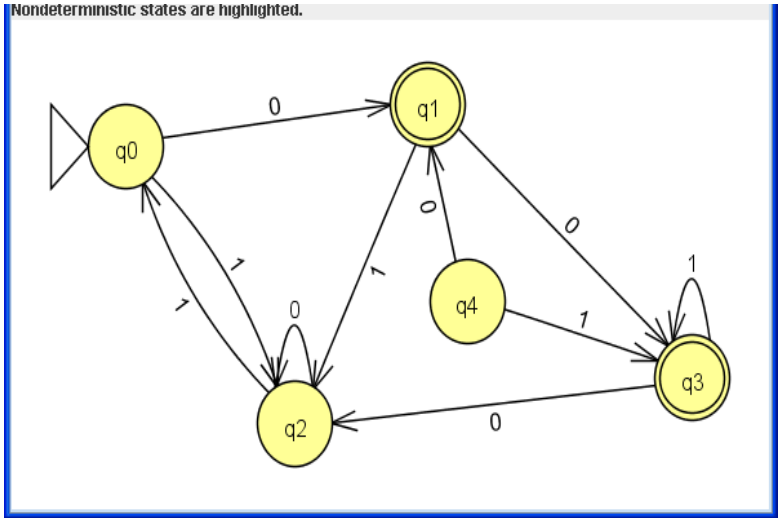
Lauf für das Wort $w = w_1w_2...w_n$

Beginnt der Lauf im Anfangszustand und
endet er in einem Endzustand,
so wird das zugehörige Wort akzeptiert

- Ausdehnung von δ auf Worte

δ^*

- $\delta^* : Q \times \Sigma^* \rightarrow Q$ definiert durch
  - $\delta^*(q, \varepsilon) = q$ // leeres Wort
  - $\delta^*(q, a.u) = \delta^*(\delta(q,a),u))$ // w = a.u
- A akzeptiert w $\Leftrightarrow \delta^*(q_0, w) \in F$
- Ein Zustand q heißt erreichbar, falls es ein Wort w gibt mit $\delta^*(q_0, w) = q$



$\delta^*(q_0,1100) = q_3$
$\delta^*(q_4,1010) = q_1$

q_4 ist nicht erreichbar

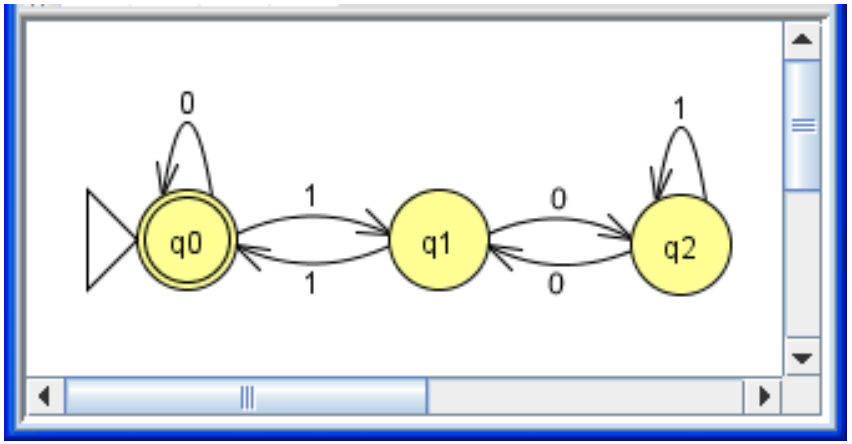
Sprache eines Automaten

Die Sprache eines Automaten ist die Menge aller Worte, die er akzeptiert. Formal:

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

- Beobachtungen
  - $\varepsilon \in L(A) \Leftrightarrow q_0 \in F$
  - nicht erreichbare Zustände können entfernt werden. Für den restlichen Automat A‘ gilt :

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



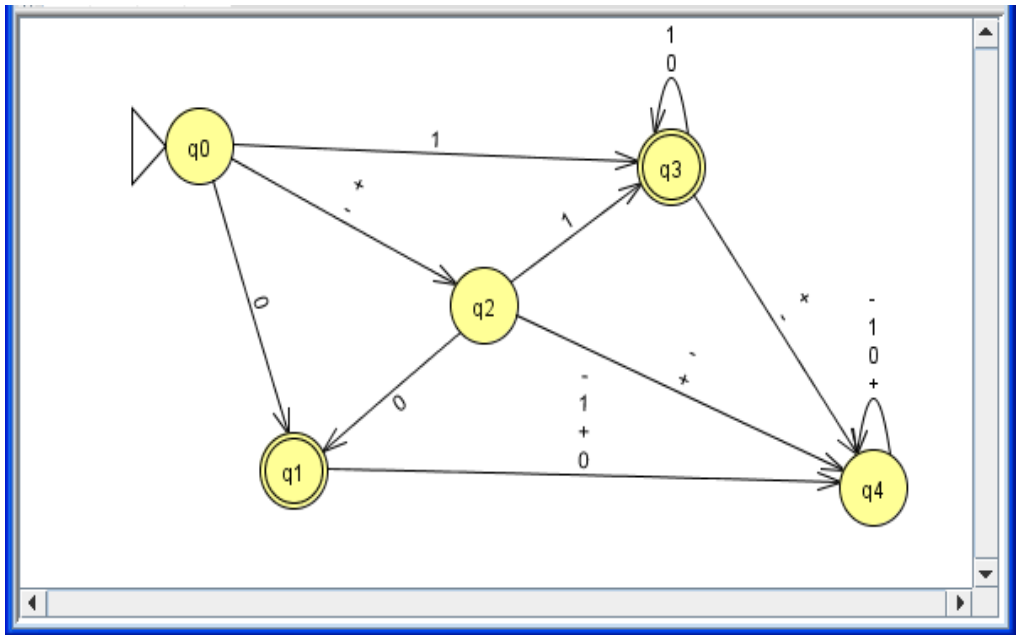
Beispiel: $w \in L(A) \Leftrightarrow$

$w = \varepsilon$ oder $(w)_2 \bmod 3 = 0$

Implementierung

- Automaten sind leicht zu implementieren
- $\delta : Q \times \Sigma \rightarrow Q$ als Tabelle
- $F \subseteq Q$
- $q_0 \in Q$

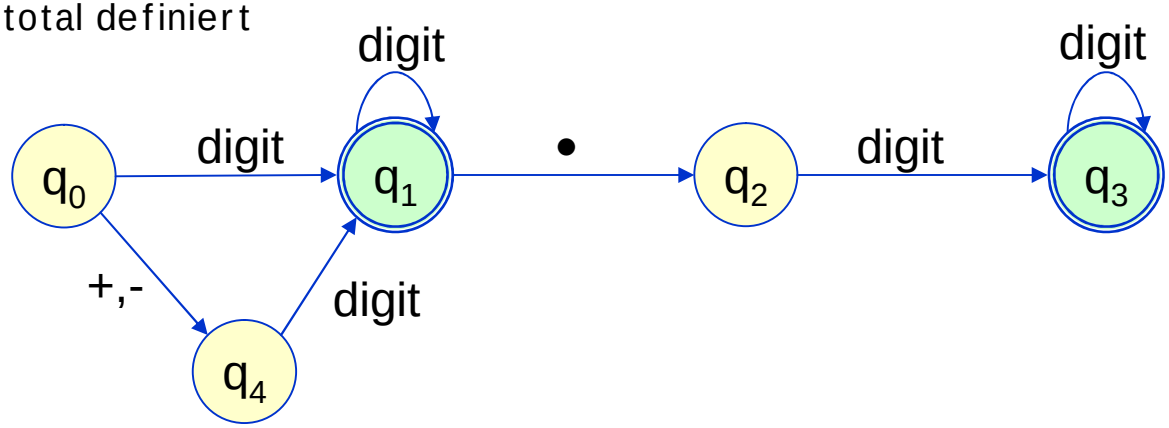
$F = \{ q_1, q_3 \}$
init = q_0.



δ	0	1	+	-
q_0	q_1	q_3	q_2	q_2
q_1	q_4	q_4	q_4	q_4
q_2	q_1	q_3	q_4	q_4
q_3	q_3	q_3	q_4	q_4
q_4	q_4	q_4	q_4	q_4

Vervollständigung

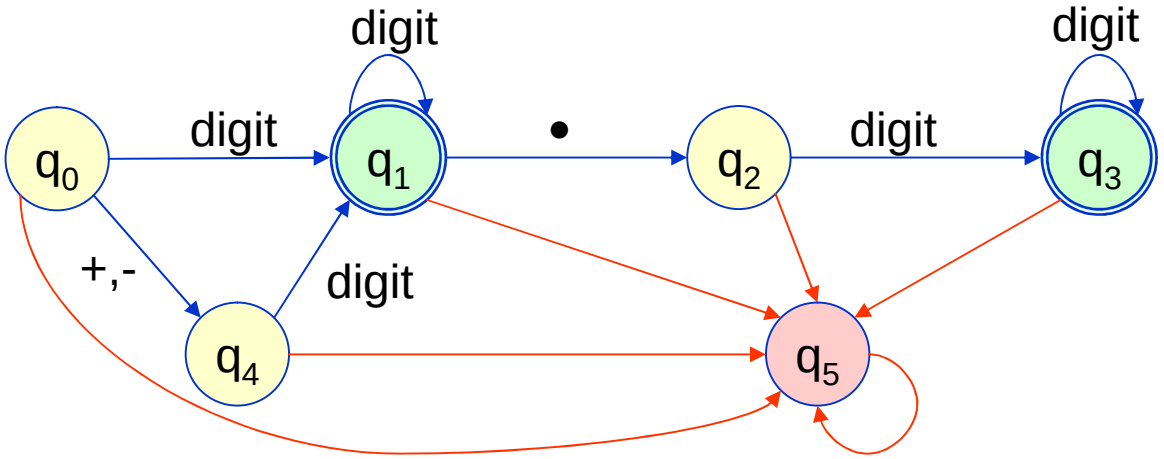
- Noch kein DFA
  - δ nicht total definiert



δ	+	-	Ziffer	.
q_0	q_4	q_4	q_1	
q_1			q_1	q_2
q_2			q_3	
q_3			q_3	
q_4			q_1	

	Endzustand?
q_0	false
q_1	true
q_2	false
q_3	true
q_4	false

Vervollständigung durch Fangzustand



δ	+	-	Ziffer	.
q_0	q_4	q_4	q_1	q_5
q_1	q_5	q_5	q_1	q_2
q_2	q_5	q_5	q_3	q_5
q_3	q_5	q_5	q_3	q_5
q_4	q_5	q_5	q_1	q_5
q_5	q_5	q_5	q_5	q_5

	Endzustand?
q_0	false
q_1	true
q_2	false
q_3	true
q_4	false
q_5	false

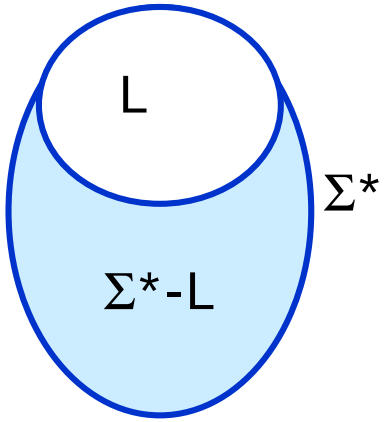
Komplementautomat

- Zu jedem Automaten $A = (Q,\Sigma,\delta,q_0,F)$ gibt es einen Automaten

  $\bar{A}$ mit $L(\bar{A}) = \Sigma^* - L(A)$.

  - Beweis: Wähle $\bar{A} := (Q,\Sigma,\delta,q_0,Q-F)$. Dann gilt:

    - $w \in L(\bar{A})$ $\Leftrightarrow \delta^*(q_0,w) \in Q-F$ // Def. $\bar{A}$
      $\Leftrightarrow \delta^*(q_0,w) \notin F$
      $\Leftrightarrow w \notin L(A)$

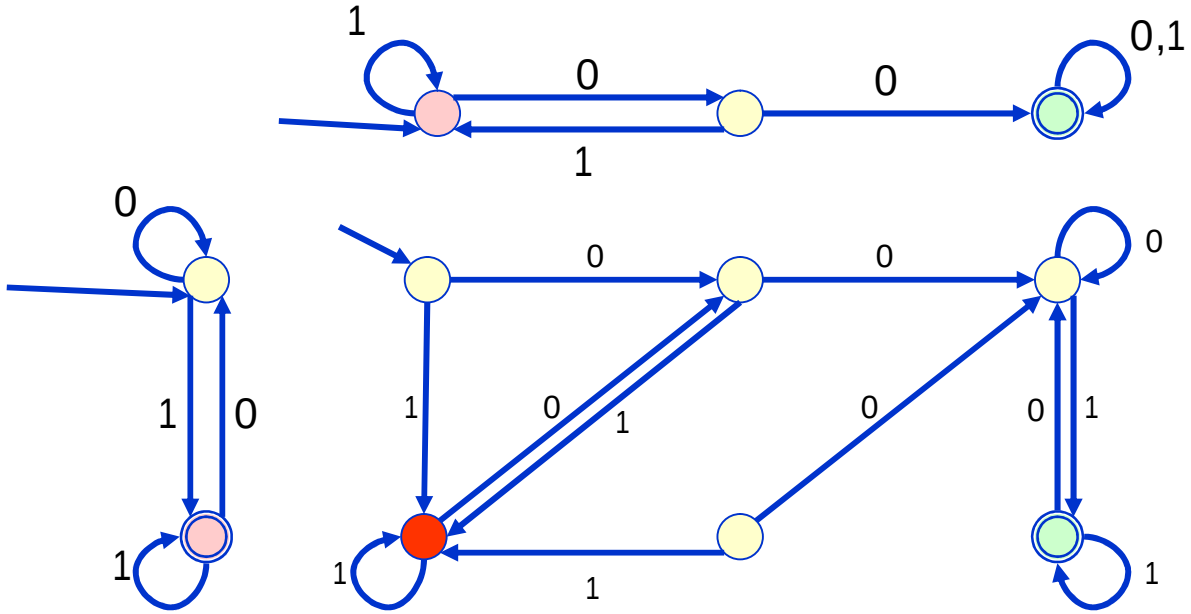


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$ seien Automaten

- $A\times B = (P\times Q, \Sigma, \delta_{A\times B}, (p_0,q_0), F_A\times F_B)$ // 1.Komp. in F_A, 2. in F_B

  $\delta_{A\times B}((p,q),a) := (\delta_A(p,a), \delta_B(q,a))$ // komponentenweise



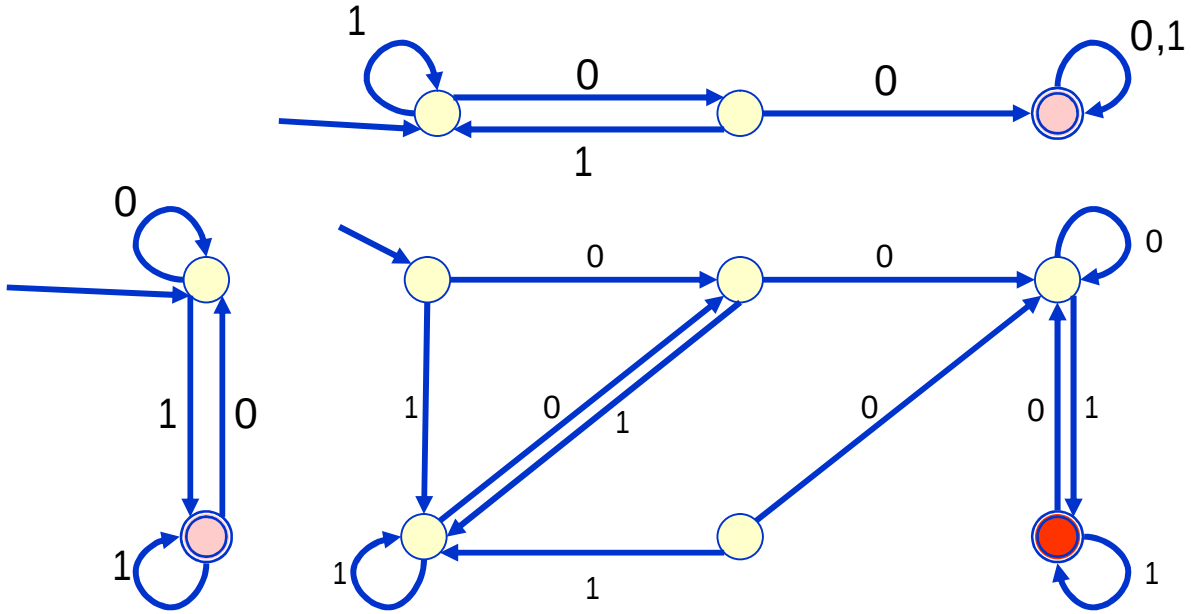
1	0	1	1	0	0	1	0	1

Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$ seien Automaten

- $A\times B = (P\times Q, \Sigma, \delta_{A\times B}, (p_0,q_0), F_A\times F_B)$ // 1.Komp. in F_A, 2. in F_B

  $\delta_{A\times B}((p,q),a) := (\delta_A(p,a), \delta_B(q,a))$ // komponentenweise



Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$ seien Automaten

- $A\times B = (P\times Q, \Sigma, \delta_{A\times B}, (p_0,q_0), F_A\times F_B)$ // 1.Komp. in F_A, 2. in F_B

  $\delta_{A\times B}((p,q),a) := (\delta_A(p,a), \delta_B(q,a))$ // komponentenweise

- Lemma: $\delta_{A\times B}^*((p,q),w) = (\delta_A^*(p,w), \delta_B^*(q,w))$ für alle $w\in\Sigma^*$.

  Beweis: (Übung, Induktion über w)

- $L(A\times B) = L(A) \cap L(B)$

  Beweis: $w\in L(A\times B) \Leftrightarrow \delta_{A\times B}^*((p_0,q_0),w) \in F_A\times F_B$ // Def. $F_{A\times B}$

  $\Leftrightarrow (\delta_A^*(p_0,w), \delta_B^*(q_0,w)) \in F_A\times F_B$ // Lemma

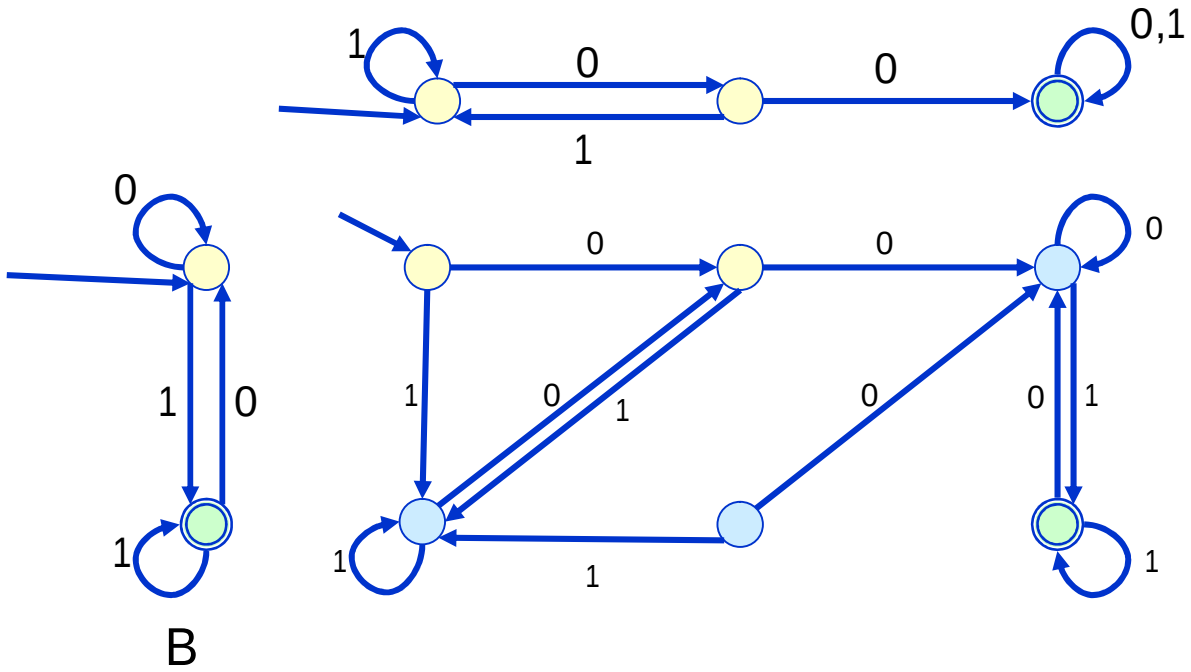
  $\Leftrightarrow \delta_A^*(p_0,w) \in F_A$ und $\delta_B^*(q_0,w) \in F_B$ // komponentenweise

  $\Leftrightarrow w \in L(A) \cap L(B)$ // Def. L(A), L(B), $\cap$

Parallelität und Produktautomat

- Parallele Komposition
  - A und B laufen gleichzeitig
  - synchron

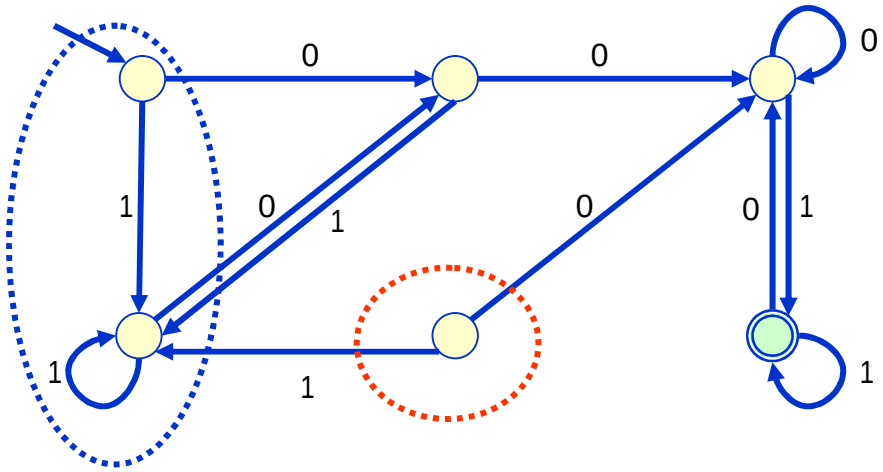
- Akzeptiere Input, falls
  - beide Automaten in Endzustand ⇒ $L(A) \cap L(B)$
  - mind. ein Automat in Endzustand : ⇒ $L(A) \cup L(B)$
    - $F=(F_A \times Q) \cup (P \times F_B)$



Produktautomat

Produktkonstruktion liefert nicht unbedingt den einfachsten Automat

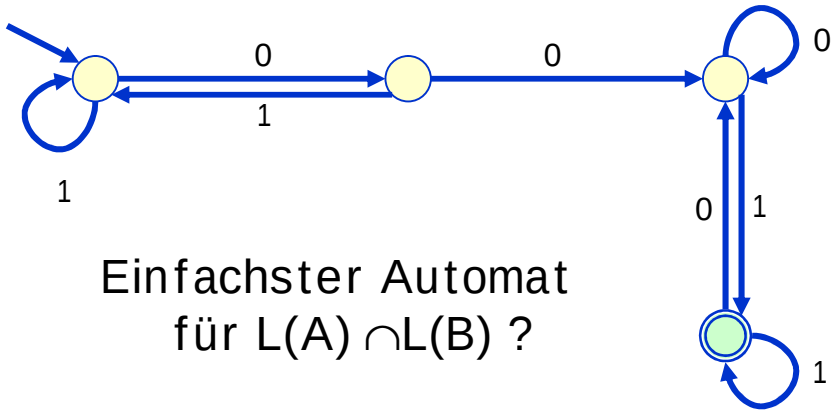
- Hier:
  - ein Zustand nicht erreichbar - entfernen
  - zwei Zustände verhaltensgleich



Produktautomat

Produktkonstruktion liefert nicht unbedingt den einfachsten Automat

- Hier:
  - ein Zustand nicht erreichbar - entfernen
  - zwei Zustände verhaltensgleich - verschmelzen

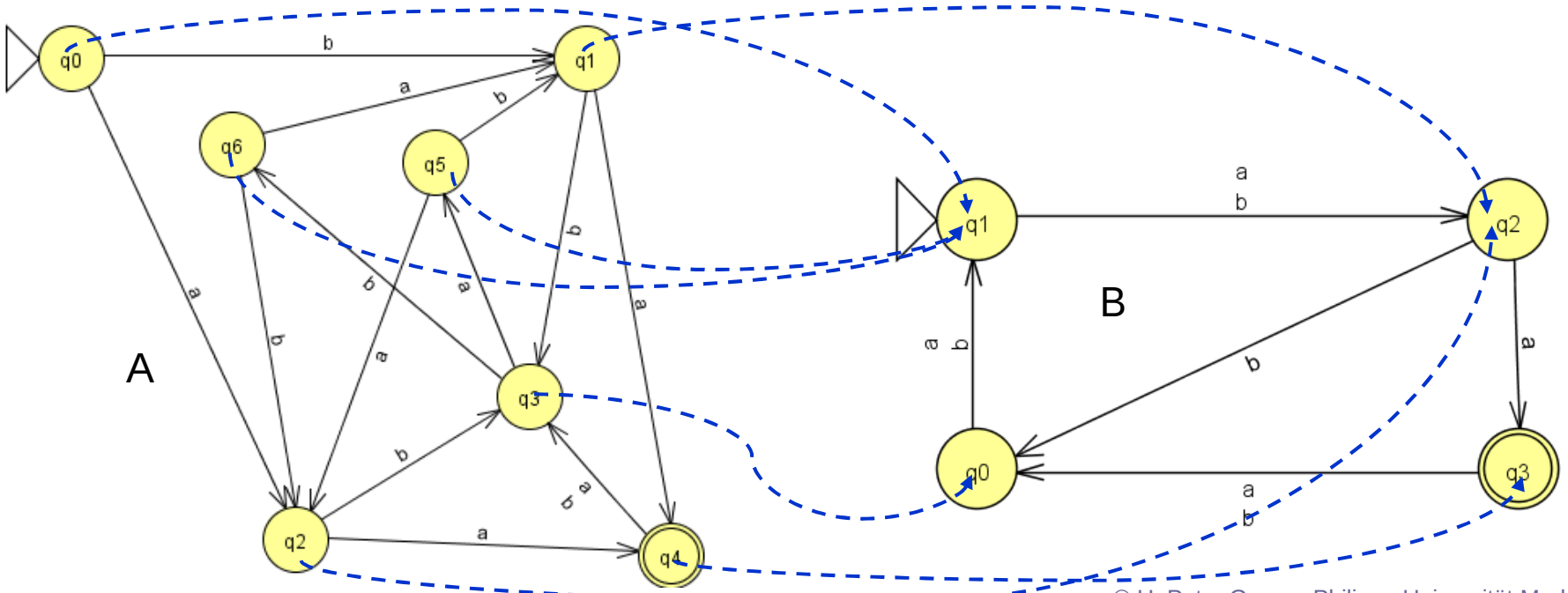


Einfachster Automat
für L(A) ∩L(B) ?

Homomorphismus

$A=(Q_A,\Sigma,\delta_A,q_A,F_A)$ und $B=(Q_B,\Sigma,\delta_B,q_B,F_B)$ Automaten
Abbildung φ: $Q_A \rightarrow Q_B$ heißt Homomorphismus, falls

1. $\varphi(q_A) = q_B$ // φ erhält Startzustand
2. $\forall q \in Q_A$: $q \in F_A \Leftrightarrow \varphi(q) \in F_B$ // φ erhält Endzustände
3. $\forall q \in Q_A: \forall a \in \Sigma$: $\varphi(\delta_A(q,a)) = \delta_B(\varphi(q),a)$ // φ erhält Transitionen



Homomorphie $\Rightarrow$ Sprachäquivalenz

Ist $\varphi : A \to B$ ein Homomorphismus, dann gilt L(A)=L(B)

Lemma: Es gilt $\varphi(\delta_A^*(q,w)) = \delta_B^*(\varphi(q),w)$.

Beweis (Übung) Induktion über Aufbau von w.

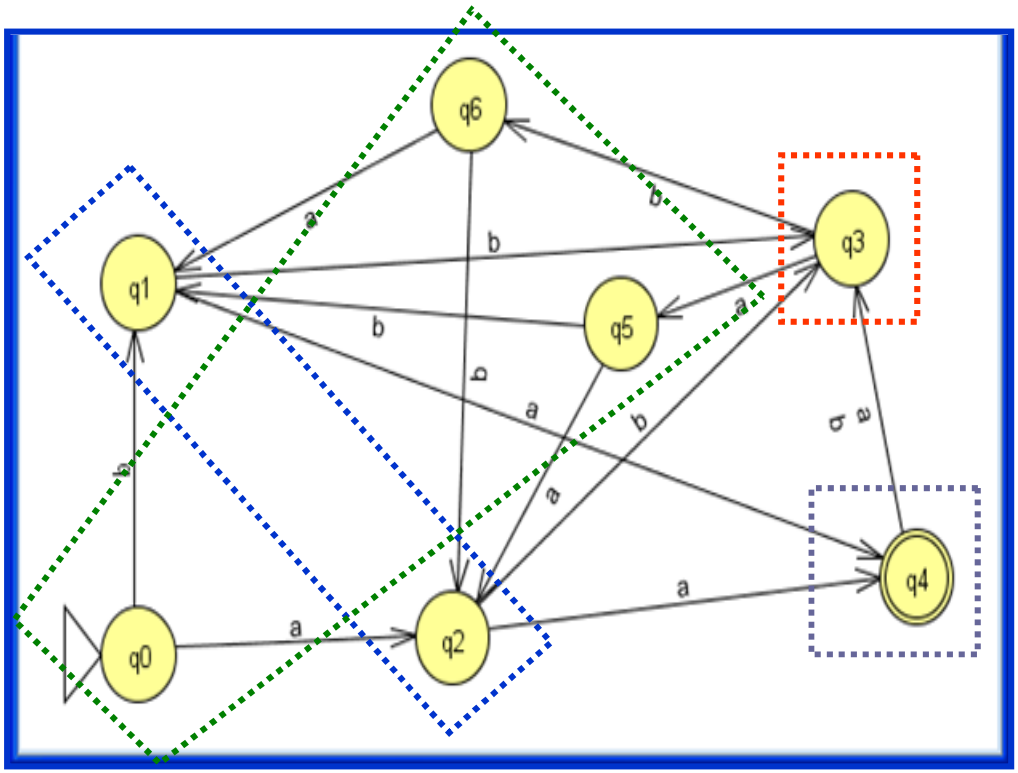
Beweis des Satzes:

$w \in L(A)$	$\Leftrightarrow \delta_A^*(q_A,w) \in F_A$	// Def. von L(A)
	$\Leftrightarrow \varphi(\delta_A^*(q_A,w)) \in F_B$	// φ Homomorph.
	$\Leftrightarrow \delta_B^*(\varphi(q_A),w) \in F_B$	// Lemma.
	$\Leftrightarrow \delta_B^*(q_B,w) \in F_B$	// φ Homomorph.
	$\Leftrightarrow w \in L(B)$	// Def. von L(B)

Kongruenzen

- Eine Kongruenzrelation Θ auf $A=(Q,\Sigma,\delta,q_0,F)$ ist
  - eine Äquivalenzrelation auf Q
  - mit $\forall (p,q) \in \Theta$: 1. $(p \in F \Leftrightarrow q \in F)$
    2. $\forall a \in \Sigma$: $\delta(p,a)\ \Theta\ \delta(q,a)$

Äquivalenzklassen: $[p]\Theta := \{ q \in Q \mid p\ \Theta\ q\}$



Beispiel:

Im gezeigten Automaten definiert die Klasseneinteilung

$\{\{q_0,q_5,q_6\},\{q_1,q_2\},\{q_3\},\{q_4\}\}$

eine Kongruenzrelation.

Aufgaben

- Homomorphismen erhalten Läufe:
  - Ist $\varphi : A \to B$ Homomorphismus, dann gilt auch für alle $q \in Q_A$ und alle w in Σ^*:

    $\varphi\, \delta_A^*(q,w) = \delta_B^*(\varphi(q),w)$
  - Hinweis: Induktion über Aufbau von w.

- Komposition von Homomorphismen:
  - $A=(Q_A,\Sigma,\delta_A,q_A,F_A)$, $B=(Q_B,\Sigma,\delta_B,q_B,F_B)$ und $C=(Q_C,\Sigma,\delta_C,q_C,F_C)$ Automaten, $\varphi : A \to B$ und $\psi : B \to C$ Homomorphismen.

    Dann ist die Hintereinanderausführung
    $$\psi \circ \varphi : A \to C$$
    ein Homomorphismus

- Ker φ ist Kongruenz:

  $\varphi : A \to B$ sei Homomorphismus, dann ist
  $\mathrm{Ker}\ \varphi = \{ (p,q) \in Q_A : \varphi(p) = \varphi(q) \}$
  eine Kongruenzrelation.

- Fortsetzung auf Worte:
  - Ist Θ Kongruenz, dann gilt für alle $w \in \Sigma^*$:
    $p \,\Theta\, q \Rightarrow \delta^*(p,w)\ \Theta\ \delta^*(q,w)$

Faktorautomat

Sei Θ eine Kongruenzrelation auf $A=(Q, \Sigma, \delta, q_0, F)$, definiere:

$$A/\Theta := (Q/\Theta, \Sigma, \delta_\Theta, q_\Theta, F_\Theta)$$

mit

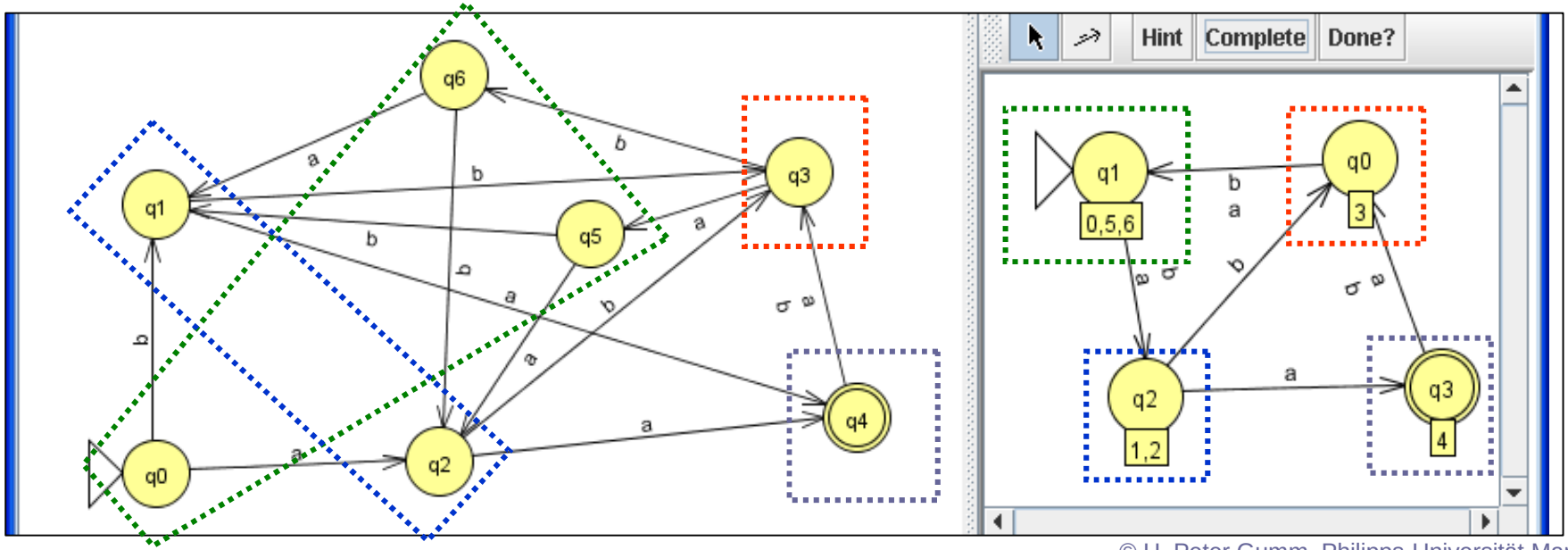
$$Q/\Theta = \{ [q]\Theta \mid q \in Q \}$$

$$q_\Theta := [q_0]\Theta$$

$$F_\Theta := \{ [q_f]\Theta \mid q_f \in F \}$$ // wohldefiniert, weil Θ Kongruenz

und $\delta_\Theta([q]\Theta, a) := [\, \delta(q,a) \,]\Theta$ // repräsentantenweise.

// wohldefiniert, da Θ Kongruenz



Faktorautomat

Sei Θ eine Kongruenzrelation auf $A=(Q, \Sigma, \delta, q_0, F)$, definiere:

$$A/\Theta := (Q/\Theta, \Sigma, \delta_\Theta, q_\Theta, F_\Theta)$$

mit

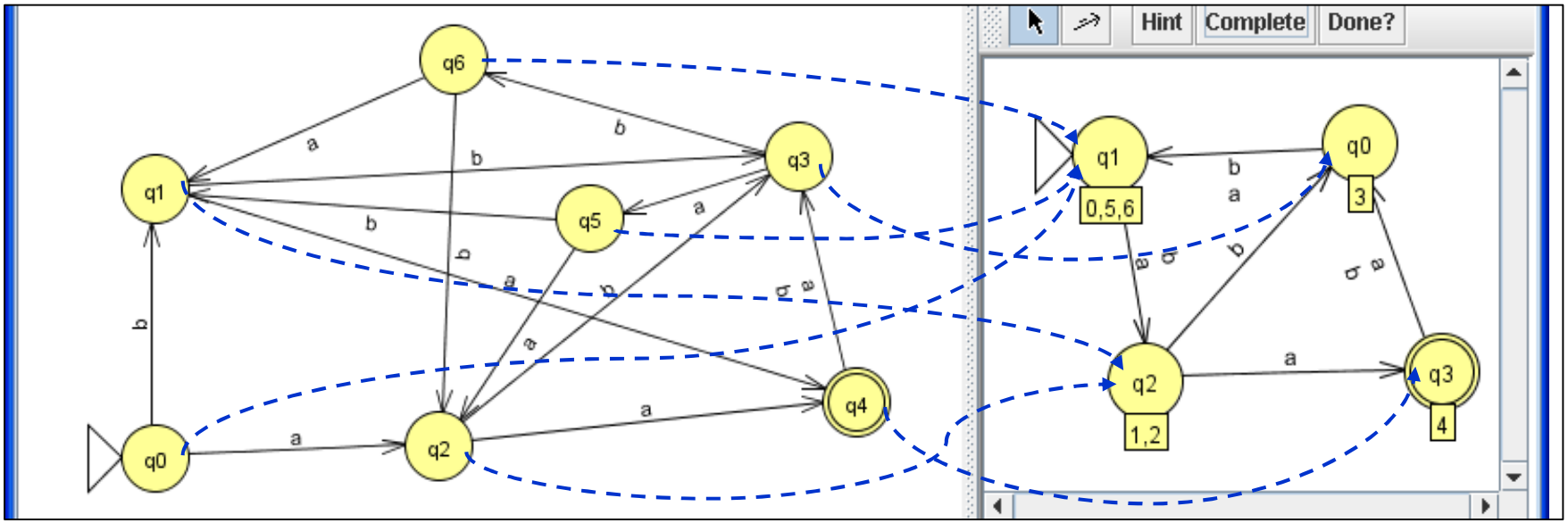
$$Q/\Theta = \{ [q]\Theta \mid q \in Q \}$$

$$q_\Theta := [q_0]\Theta$$

$$F_\Theta := \{ [q_f]\Theta \mid q_f \in F \}$$ // wohldefiniert, weil Θ Kongruenz

und $\delta_\Theta([q]\Theta, a) := [\ \delta(q,a)\ ]\Theta$ // repräsentantenweise.

// wohldefiniert, da Θ Kongruenz



Homomorphiesatz

Satz: A Automat, Θ Kongruenz. Dann definiert

$$\pi_\Theta : A \to A/\Theta \quad \text{mit} \quad \pi_\Theta(q) := [q]\Theta$$

einen Homomorphismus und es gilt $\Theta = \ker \pi_\Theta$.

Beweis: Wir rechnen die Homomorphieregeln nach

1. // π_Θ erhält Startzustand:

$$\begin{aligned} \pi_\Theta(q_0) &= [q_0]\Theta && \text{// Definition } \pi_\Theta \\ &= q_\Theta && \text{// Def. von Startzustand in } A/\Theta \end{aligned}$$

2. // π_Θ erhält Endzustände

$$\begin{aligned} q \in F_A &\Leftrightarrow [q]\Theta \in F_\Theta && \text{// Def. Endzustände in } A/\Theta \\ &\Leftrightarrow \pi_\Theta(q) \in F_\Theta && \text{// Definition } \pi_\Theta \end{aligned}$$

3. // π_Θ erhält Transitionen

$$\begin{aligned} \pi_\Theta(\delta_A(q,a)) &= [\delta_A(q,a)]\Theta && \text{// Def. } \pi_\Theta \\ &= \delta_\Theta([q]\Theta,a) && \text{// Def. von } \delta_\Theta \text{ in } A/\Theta \end{aligned}$$

Schließlich:

$$(p,q) \in \ker \pi_\Theta \Leftrightarrow \pi_\Theta(p) = \pi_\Theta(q) \Leftrightarrow [p]\Theta = [q]\Theta \Leftrightarrow (p,q) \in \Theta$$

Inhalt

1. Deterministische Akzeptoren
   - Grundbegriffe
   - Sprache eines Automaten
   - Implementierung
   - Komplement, Produkte
   - Homomorphismen von Automaten
   - Faktorautomat
   - Homomorphiesatz

2. Minimierung und Grenzen von Automaten

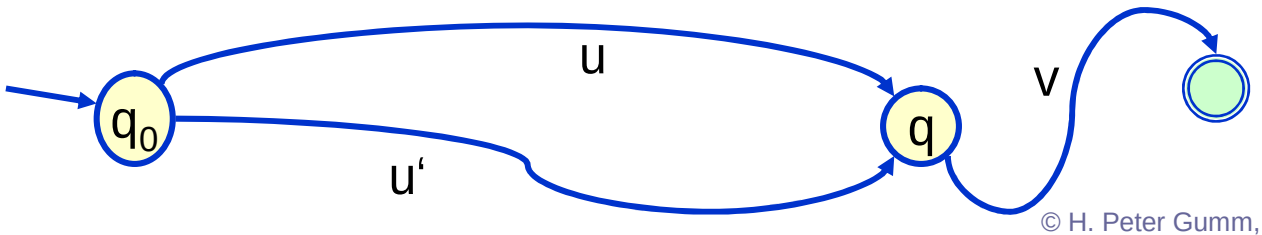
   - Trennbarkeit
   - Nerode-Lemma
   - Pumping Lemma
   - nicht reguläre Sprachen
   - Minimierung

Ziel : Minimierung

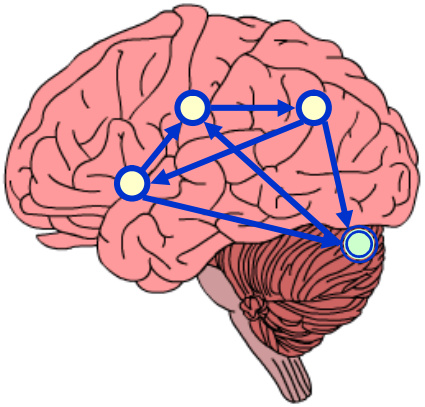
- A Automat, Θ Kongruenz. Dann gilt $L(A) = L(A/\Theta)$.
  - Folgt aus dem vorigen Satz.
  - Wenn Θ nicht trivial, dann hat A/Θ weniger Zustände als A
  - Gesucht: Θ, so dass für möglichst viele $p,q \in Q$ gilt: $p \Theta q$.
- Gesucht: Θ, so dass für möglichst viele $p,q \in Q$: $p \Theta q$.
  - Es muss da eine Grenze geben …
- Kriterium für $(p,q) \notin \Theta$?
  - $\delta^*(p,w) \in F$, $\delta^*(q,w) \notin F \Rightarrow (p,q) \notin \Theta$ // 2. Kongr.Bed.

Das „*Gedächtnis*“ von Automaten

- Jetzt menschelt es
  - keine Angst, wir werden bald wieder präziser
- Angenommen, Automat A „will“ ein Wort w erkennen.
  - Nachdem Anfangsteil u eingelesen ist, befindet er sich im Zustand $q = \delta^*(q_0,u)$.
  - Ob er den Rest v, und damit $w = uv$ akzeptiert, hängt nur vom Zustand q ab.
- q ist die einzige Information, die sich der Automat „merken“ kann, nicht aber, wie er zu q gekommen ist.
  - Konkret:
    Wenn $w = uv \in L$ und $\delta^*(q_0,u) = q = \delta^*(q_0,u')$,
    dann muss auch $w' = u'v \in L$



DFA‘s haben endliches Gedächtnis



- Sehr sympathisch
  - kennen wir alle
  - aber was bedeutet das genau ?
- Beispiel: A soll $L_{anbn} = \{ a^n b^n \mid n \geq 0 \}$ erkennen.
- Intuitiv:
  - Geht nicht, denn nachdem k viele a‘s eingelesen sind, müsste er sich Automat k „gemerkt“ haben, um nach der richtigen Anzahl von b‘s in einen Endzustand zu gehen.
  - Er kann sich aber nur endlich viele verschiedene Informationen merken.
    - meint: Er kann nicht für jedes $k \in$ Nat in einem anderen Zustand sein
- mathematisch:
  - $\exists\, i \neq k: \delta^*(q_0,a^i) = q = \delta(q_0,a^k) = q.$
    - einerseits : $\delta^*(q,b^i) = \delta^*(\delta^*(q_0,a^i),b^i) = \delta^*(q_0,a^ib^i) \in F$ sein
    - andererseits : $\delta^*(q,b^i) = \delta^*(\delta^*(q_0,a^k),b^i) = \delta^*(q_0,a^kb^i) \notin F$ sein
  - Widerspruch !
  - Also gibt es keinen endlichen Automaten, der L_{anbn} erkennt

Intuition liefert Anfangsverdacht

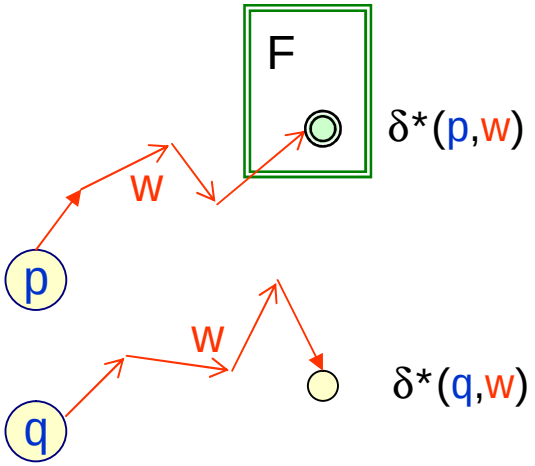
- Mit der Intuition kann man gut fundierte Vermutungen anstellen, z.B. dass folgende Sprachen nicht von endlichen Automaten akzeptiert werden können:
  - $L = \{ a^i b^k \mid i \neq k \}$ nicht L(A) für DFA A
    - Nach i vielen a's müsste A sich deren Anzahl gemerkt haben, damit er i viele b's nicht akzeptiert wohl aber jede andere Anzahl
  - $L = \{ u^R u \mid u \in \Sigma^* \}$ Palindrome
  - L = Menge aller wohlgeformten Klammerausdrücke
  - $L = \{ a^{n*n} \mid n \geq 0 \}$
    - klein bisschen schwieriger zu argumentieren

- Die Intuition liefert aber nur Anfangsverdacht
- Wir lernen zwei mathematische Beweismethoden kennen:
  - Nerode Lemma
  - Pumping Lemma

Trennbare Zustände

Zustände p und q heißen trennbar, wenn es ein Wort w gibt mit $\delta^*(p,w) \in F$ aber $\delta^*(q,w) \notin F$, oder umgekehrt $\delta^*(p,w) \notin F$ und $\delta^*(q,w) \in F$.

- Zustände p, q sind also nicht trennbar wenn
  - $\forall w \in \Sigma^*: (\delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F)$
- Definition: $p \sim_A q :\Leftrightarrow$ p und q sind nicht trennbar.
- $\sim_A$ ist eine Äquivalenzrelation:
  - $p \sim_A p$ - klar
  - $p \sim_A q \Leftrightarrow \forall w \in \Sigma^*: (\delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F)$
    $\Leftrightarrow \forall w \in \Sigma^*: (\delta^*(q,w) \in F \Leftrightarrow \delta^*(p,w) \in F)$
    $\Leftrightarrow q \sim_A p$
  - $p \sim_A q \wedge q \sim_A r \Rightarrow p \sim_A r$ - analog



$\sim_A$ ist Kongruenz

Auf jedem Automaten $A=(Q, \Sigma, \delta, q_0, F)$ ist $\sim_A$ eine Kongruenzrelation.

Beweis: $\sim_A$ ist Äquivalenzrelation – klar.

Kongruenz:

$q \sim_A q' \Rightarrow \forall w \in \Sigma^*: (\delta^*(q,w) \in F \Leftrightarrow \delta^*(q',w) \in F)$ //Def. von ~

$\Rightarrow \forall a \in \Sigma: \forall v \in \Sigma^*: (\delta^*(q,av) \in F \Leftrightarrow \delta^*(q',av) \in F)$ // Setze w=av

$\Rightarrow \forall a \in \Sigma: \forall v \in \Sigma^*: (\delta^*(\delta(q,a),v) \in F \Leftrightarrow \delta^*(\delta(q',a),v) \in F)$ // Def. δ^*

$\Rightarrow \forall a \in \Sigma: \delta(q,a) \sim_A \delta(q',a)$ //Def. von ~

Folgerung:

$L(A) = L(A/\sim)$

$\sim_A$ ist größte Kongruenz

R sei Relation mit

1. $p\ R\ q \Rightarrow (p \in F \wedge q \in F) \vee (p \notin F \wedge q \notin F)$ d.h. $(p \in F \Leftrightarrow q \in F)$
2. $p\ R\ q \Rightarrow \forall a \in \Sigma: \delta(p,a)\ R\ \delta(p,a)$

dann gilt $R \subseteq \sim_A$

Beweis: Sei $p\ R\ q$. Wir müssen zeigen:

$\forall w \in \Sigma^*: (\forall p,q \in Q: p\ R\ q \Rightarrow \delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F)$

Induktionshyp.: P(w): $\forall p,q \in Q: p\ R\ q \Rightarrow (\delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F)$

Induktionsanfang $w = \varepsilon$:

$\delta^*(p,\varepsilon) \in F \Leftrightarrow p \in F \Leftrightarrow q \in F \Leftrightarrow \delta^*(q,\varepsilon) \in F.$ // Def δ^* u. Ax. 1 für R

Induktionsschritt $w = a.v$:

Sei $\forall p,q \in Q.\ p\ R\ q \Rightarrow (\delta^*(p,v) \in F \Leftrightarrow \delta^*(q,v) \in F)$ // die Ind.hypothese

dann $\delta^*(p,a.v) \in F \Leftrightarrow \delta^*(\delta(p,a),v) \in F$ // Def. δ^*

$\Leftrightarrow \delta^*(\delta(q,a),v) \in F$ // Ax. 2 für R u. Ind.Hyp. // für $\delta(p,a)$, $\delta(q,a)$, v

$\Leftrightarrow \delta^*(q,a.v) \in F$ // Def. δ^*

Folgerungen: $\sim_A$ ist größte Relation mit den Axiomen 1. und 2.

Jede Kongruenz Θ ist in $\sim_A$ enthalten.

Zustände in $A/\sim_A$ sind trennbar

In $A/\sim_A$ sind je zwei verschiedene Zustände trennbar.

Beweis: Gegeben $[p]\sim$ und $[q]\sim$ aus $A/\sim$. (Wir schreiben $\sim$ statt $\sim_A$)

$[p]\sim \neq [q]\sim \quad \Rightarrow \neg (p \sim q)$ // Def [ ]~

(o.B.d.A.) $\Rightarrow \exists\, w \in \Sigma^*: (\delta^*(p,w) \in F \wedge \delta^*(q,w) \notin F)$ // Def ~

$\Rightarrow \pi_\sim (\delta^*(p,w)) \in F_\sim \wedge \pi_\sim (\delta^*(q,w)) \notin F_\sim$ // $\pi_\sim$ Homo.

$\Rightarrow \delta_\sim^*(\pi_\sim(p),w) \in F_\sim \wedge \delta_\sim^*(\pi_\sim(q),w) \notin F_\sim$ // $\pi_\sim$ Homo.

$\Rightarrow \delta_\sim^*([p]\sim,w) \in F_\sim \wedge \delta_\sim^*([q]\sim,w) \notin F_\sim$ // Def π_Θ

$\Rightarrow$ $[p]\sim$ und $[q]\sim$ trennbar // Def. trennbar

Zusammenfassung

Zu jedem Automaten A gibt es einen Automaten $A_{\sim}$ mit

- jeder Zustand von $A_{\sim}$ ist erreichbar
- je zwei Zustände von $A_{\sim}$ sind trennbar
- $L(A) = L(A_{\sim})$

Beweis:

Entferne von A alle nichterreichbaren Zustände. Für den entstandenen Automaten $\hat{A}$ gilt: $L(A)=L(\hat{A})$.

Setze $A_{\sim} := \hat{A}/\sim_{\hat{A}}$. Verifiziere:

- Jeder Zustand in $A_{\sim}$ ist erreichbar
- Je zwei Zustände in $A_{\sim}$ sind trennbar
- $L(A) = L(A_{\sim})$.

L-trennbar

Sei L eine Sprache. Zwei Worte $u, v \in \Sigma^*$ heißen L-trennbar, falls es ein $w \in \Sigma^*$ gibt mit $uw \in L$ aber $vw \notin L$, oder umgekehrt.

Beispiel: L = { 0, 010, 0110, 01110 }

- u = 0 und v = 01 sind L-trennbar mit Hilfe von w = 0,
  denn $00 \notin L$ aber $010 \in L$
- 01 und 011 sind nicht L-trennbar.

Beobachtung: Sind $u, v \in \Sigma^*$ L-trennbar mittels $w \in \Sigma^*$, dann muss

- w ein Suffix
- u oder v ein Präfix

eines Wortes aus L sein

L-trennbar $\cong$ A-trennbar

L Sprache:

$u, v \in \Sigma^*$ L-trennbar

$\Leftrightarrow \exists w \in \Sigma^*: (uw \in L \wedge vw \notin L) \vee (uw \notin L \wedge vw \in L)$

A Automat:

$p,q \in Q_A$ A-trennbar

$\Leftrightarrow \exists w \in \Sigma^*: (\delta(p,w) \in F_A \wedge \delta(q,w) \notin F_A) \vee (\delta(p,w) \notin F_A \wedge \delta(q,w) \in F_A)$

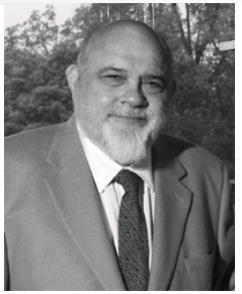
Lemma: Ist L eine Sprache und A ein Automat mit L=L(A).
Zwei Worte $u, v \in \Sigma^*$ sind genau dann L-trennbar, wenn
die Zustände $\delta^*(q_0,u)$ und $\delta^*(q_0,v)$ A-trennbar sind.

Beweis: u und v seien L-trennbar, dann gibt es $w \in \Sigma^*$ mit
$uw \in L$ aber $vw \notin L$

$\Leftrightarrow \delta^*(q_0, uw) \in F$ aber $\delta^*(q_0, vw) \notin F$ //L=L(A)

$\Leftrightarrow \delta^*(\delta^*(q_0, u),w) \in F$ aber $\delta^*(\delta^*(q_0, v),w) \notin F$

Nerode-Lemma



Anil Nerode
*1956

Nerode: L sei eine Sprache. Gibt es n Worte, die paarweise L-trennbar sind, so hat jeder Automat, der L erkennt, mindestens n verschiedene Zustände.

- Beweis:

  $w_1, \ldots, w_n \in \Sigma^*$ paarweise L-trennbar und L=L(A) // Voraussetzung

  $\Rightarrow \delta^*(q_0,w_1), \ldots, \delta^*(q_0,w_n)$ A-trennbar // Lemma

  $\Rightarrow \delta^*(q_0,w_1), \ldots, \delta^*(q_0,w_n)$ paarweise verschieden // p,q trennbar $\Rightarrow p \neq q$

  $\Rightarrow$ A hat mindestens n verschiedene Zustände // q.e.d.

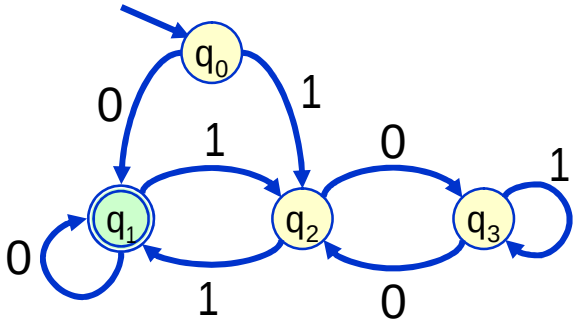
Anwendungen des Nerode-Lemmas

- L_{drei}= { 0, 00, 11, 011, 110, 1001, ... } = alle Binärzahlen, die durch 3 teilbar sind.

  - Eine trennbare Menge mit 4 Elementen ist z.B. { ε, 0, 1, 10 } :
    - ε trennt ε und 0, denn εε ∉ L_{drei} aber 0ε = 0∈ L_{drei}
    - 1 trennt ε und 1, denn ε1 ∉ L_{drei} aber 11 ∈ L_{drei}
    - 01 trennt ε und 10, denn ε01 ∉ L_{drei} aber 1001 ∈ L_{drei}
    - 0 trennt 0 und 1, denn 00 ∉ L_{drei} aber 10 ∈ L_{drei}
    - 0 trennt 0 und 10, denn 00 ∉ L_{drei} aber 100 ∈ L_{drei}
    - 1 trennt 1 und 10, denn 11 ∈ L_{drei} aber 101 ∉ L_{drei}

  - Folglich hat jeder Automat, der L_{drei} erkennt mindestens 4 Elemente

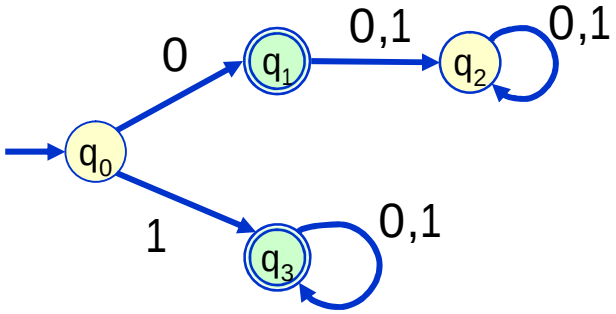
trennt	ε	0	1	10
ε		ε	1	01
0			0	0
1				1
10				



Nerode-Lemma

- L_{bin} = { 0, 1, 10, 11, 101, ... } = alle Binärzahlen ohne führende Nullen
  - Eine trennbare Menge mit vier Elementen ist z.B.: {ε, 0, 1, 01}
    - ε trennt ε und 0, denn εε ∉ L_{bin} aber 0ε ∈ L_{bin}
    - ε trennt ε und 1, denn εε ∉ L_{bin} aber 1ε ∈ L_{bin}
    - 0 trennt ε und 01, denn ε0 ∈ L_{bin} aber 010 ∉ L_{bin}
    - ε trennt 0 und 01, denn 0ε ∈ L_{bin} aber 01ε ∉ L_{bin}
    - ε trennt 1 und 01, denn 1ε ∈ L_{bin} aber 01ε ∉ L_{bin}
    - 0 trennt 0 und 1, denn 00 ∉ L_{bin} aber 10 ∈ L_{bin}
  - Folglich hat jeder Automat, der L_{bin} erkennt, mindestens 4 Zustände

trennt	ε	0	1	01
ε		ε	ε	0
0			0	ε
1				ε
01				



Produktautomat

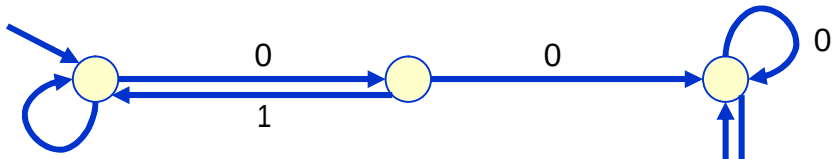
Produktkonstruktion lieferte nicht unbedingt den einfachsten Automat

- Hier: $\Sigma = \{0,1\}$
  - $L = L(A) \cap L(B) = \Sigma^*00\,\Sigma^*1$ - Teilwort 00 und letztes Zeichen
- Zeige, dass z.B. $\{\varepsilon, 0, 00, 001\}$ eine trennbare Menge für L ist:
  - Dann muss jeder Automat für L mindestens 4 Zustände haben
- Wie kommt man auf diese Menge ?
  - Wenn man schon einen Automaten mit L=L(A) hat, gilt:

  $M=\{m_1,\ldots,m_k\}$ trennbare Menge für L(A)

  $\Leftrightarrow$ $\{\delta^*(q_0,m_1), \ldots, \delta^*(q_0,m_k)\}$ paarweise unterscheidbar

trennt	ε	0	00	001
ε		01	1	ε
0			1	ε
00				ε
001				



Einfachster Automat für $\Sigma^*00\,\Sigma^*1$

?

Klammersprache nicht endlich erkennbar

- L_{kla} = { ε, (), ()(), (()), (())(), ()(()), ()()(), ... }

  = Menge aller wohlgeformten Klammerausdrücke

- Eine trennbare Menge ist z.B.: {ε, (, ((, (((, ((((, ... }, denn
  -) trennt ε und (, denn ε) ∉ L_{bin} aber () ∈ L_{bin}
  -)) trennt ε und)), denn ε)) ∉ L_{bin} aber (()) ∈ L_{bin}
  - ...

  für n≠k:
  - $)^n$ trennt $(^n$ und $(^k$, denn $(^n)^n$ ∈ L_{bin} aber $(^k)^n$ ∉ L_{bin}
  - ...

  - Folglich hat jeder Automat, der L_{kla} erkennt, unendlich viele Zustände

- Es gibt keinen endlichen Automaten, der die Sprache aller wohlgeformten Klammerausdrücke erkennt !!!

trennt	ε	(	((	(((	((((
ε		)	))	)))	))))
(			))	)))	))))
((				)))	))))
(((					))))
(((					

...

...

Palindrome nicht endlich erkennbar

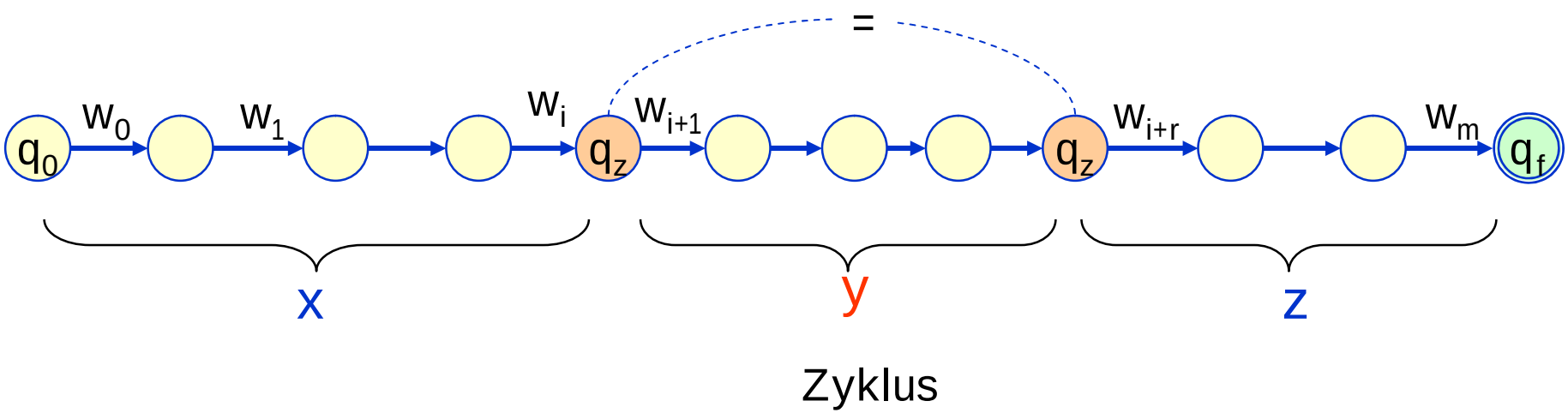
- L_{pal} = { $w^R w$ | $w \in \{a,b,c\}^*$ } = Menge aller Palindrome über $\Sigma=\{a,b,c\}$
- Eine trennbare Menge ist z.B.: { $a^n b$ | $n \geq 0$ }, denn
  - für $k > i \geq 0$ gilt:

$$a^k b b a^k \in L_{pal} \text{ aber } a^i b b a^k \notin L_{pal}$$

- Folglich gibt es keinen endlichen Automaten, der L_{pal} erkennt

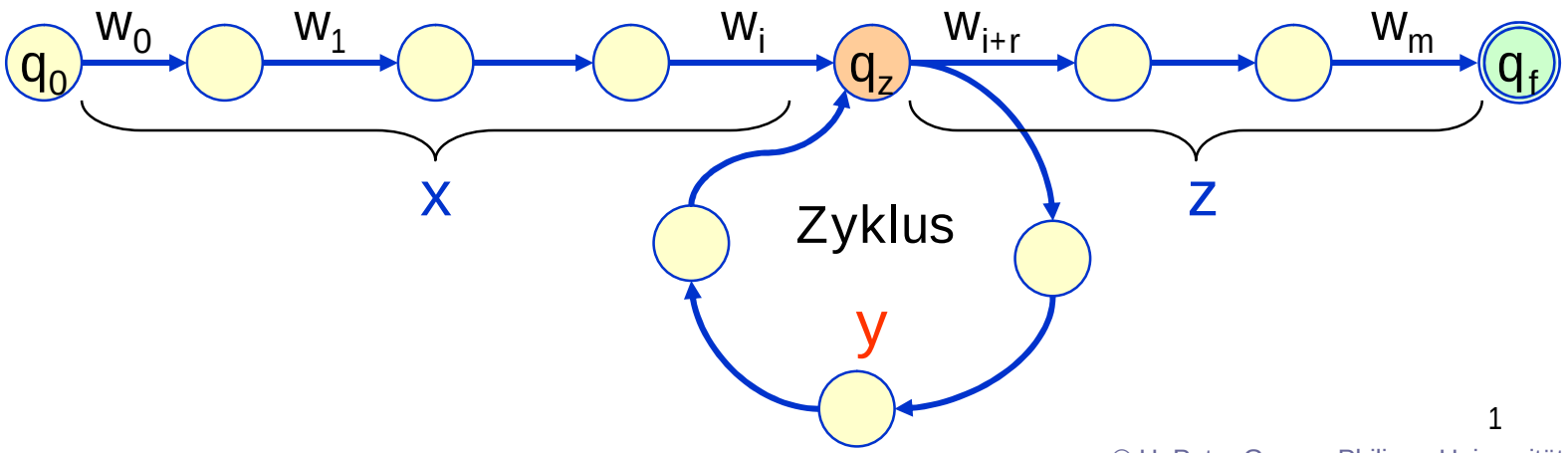
Lange Worte in kleinen Automaten

- A sei ein endlicher Automat mit k Zuständen
  - Ist $w \in L(A)$ mit $|w| \geq k$, dann hat jeder Lauf für w einen Zyklus.
    - Klar, weil der Lauf mindestens k+1 Zustände besucht.
    - Es gibt aber nur k verschiedene Zustände, also muss ein Zustand mehrfach besucht werden.
  - w lässt sich zerlegen als $w = xyz$ mit $|xy| \leq k$, $|y| \geq 1$
    - x : den Teil vor Beginn des ersten Zyklus, y : den Zyklus und z : den Rest.
  - dann sind aber auch
    $xz \in L(A)$, $xyyz \in L(A)$, $xyyyz \in L(A)$, ..., $xy^kz \in L(A)$, für alle $k \geq 0$.



Lange Worte in kleinen Automaten

- A sei ein endlicher Automat mit k Zuständen
  - Ist $w \in L(A)$ mit $|w| \geq k$, dann hat jeder Lauf für w einen Zyklus.
    - Klar, weil der Lauf mindestens k+1 Zustände besucht.
    - Es gibt aber nur k verschiedene Zustände, also muss ein Zustand mehrfach besucht werden.
  - w lässt sich zerlegen als $w = xyz$ mit $|xy| \leq k$, $|y| \geq 1$
    - x : den Teil vor Beginn des ersten Zyklus, y : den Zyklus und z : den Rest.
  - dann sind aber auch
    $xz \in L(A)$, $xyyz \in L(A)$, $xyyyz \in L(A)$, ..., $xy^nz \in L(A)$, für alle $n \geq 0$.



Pumping Lemma

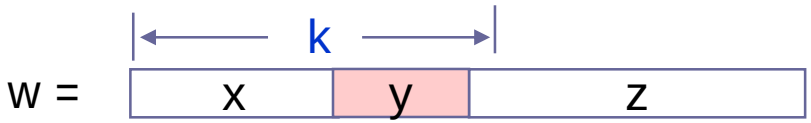
Für die Sprache L eines endlichen Automaten gibt es eine Zahl k, so dass jedes Wort $w \in L$ mit $|w| \geq k$ sich zerlegen lässt als

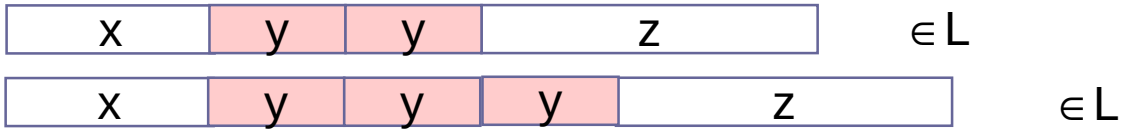
$$w = x\ y\ z$$

so dass

- $0 < |y| \leq |xy| \leq k$
- $\forall\, n \in \mathrm{Nat} : xy^n z \in L$.

Also: jedes große ($|w| \geq k$) Wort hat im vorderen Bereich ($|xy| \leq k$) ein nichtleeres Teilwort y, das sich „aufpumpen“ lässt:





etc.

Beispiel

- $L_{anbn} = \{a^n b^n \mid n \geq 0\}$ ist nicht durch endl. Automaten erkennbar.
  - Angenommen, L_{anbn} wäre regulär. Dann gäbe es ein k wie im Pumping Lemma. Jedes k-große ($|w| \geq k$) Wort $w \in L_{anbn}$ hätte im k-vorderen Bereich ($|xy| \leq k$) ein nichtleeres Teilwort y, das sich „aufpumpen“ lässt.
  - Wir betrachten jetzt das Wort $a^k b^k$
    1. es ist in L_{anbn}
    2. es ist k-gross ($|w| = 2*k > k$)
  - Es müsste im k-vorderen Bereich ein Teilwort haben, das sich aufpumpen lässt
    - der k-vordere Bereich besteht aber nur aus a‘s
    - Wenn wir hier einen nichtleeren Teil y aufpumpen, bekommen wir ein Wort mit mehr a‘s als b‘s
    - das wäre nicht mehr in L_{anbn} . Widerspruch !

| x | y | z |

aaaaaaaaaabbbbbbbbbb $\in$ L

k k

| x | y y | z |

aber aaaaaaaaaaaabbbbbbbbbb $\notin$ L

zu viele a-s

Es gibt daher keinen endlichen Automaten A mit $L_{anbn}=L(A)$

Beispiel

- $L_{fact} = \{a^{m!} \mid m \geq 3\}$ ist Sprache keines endl. Automaten
  - Du wählst ein k, // Allquantor
    - ich wähle $w = a^{k!}$ (garantiert $|w| \geq k$) // Existenzquantor
    - du zerlegst $w = a^{k!}$ als uzv mit $|uz| \leq k$, $|z| \geq 1$ // Allquantor
    - dann ist $uv \notin L_{fact}$, denn: // ich pumpe ab

sei $|z|=j \leq k$ dann ist $|uv| = k!-j > (k-1)!$ falls $k \geq 3$.

Es gibt daher keinen endlichen Automaten A mit $L_{fact}=L(A)$

Beispiel

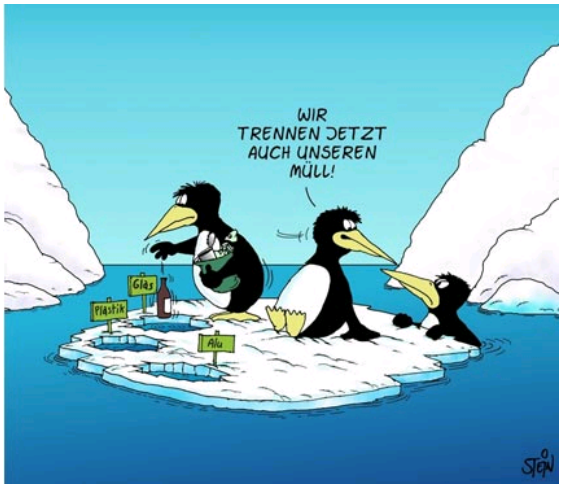
- $L_{diff} = \{a^m b^r \mid m \neq r\}$ ist nicht Sprache eines endl. Automaten
  - Pumping Lemma direkt anwenden ?
    - geht – aber schwierig
  - Besser: Wäre $L_{diff} = L(A)$, dann wäre

    $$L_{ambm} = (\Sigma^* - L_{diff}) \cap a^*b^*$$

    auch Sprache eines endl. Automaten. Widerspruch!

Es gibt daher keinen endlichen Automaten A mit $L_{diff}=L(A)$

Nerode ist meist einfacher



- $L_{diff} = \{a^m b^r \mid m \neq r\}$
  - $\{a^i \mid i \geq 0\}$ ist unendliche trennbare Menge
  - trenne a^i von allen a^j $(i \neq j)$ mittels b^i
- $L_{ambm} = \{a^m b^m \mid m \geq 0\}$
  - $\{a^i \mid i \geq 0\}$ ist unendliche trennbare Menge
  - trenne a^i von allen a^j $(i \neq j)$ mittels b^i
- $L_{fact} = \{a^{m!} \mid m \geq 0\}$
  - $\{a^{n!-n} \mid n \geq 0\}$ ist unendliche trennbare Menge
  - trenne $a^{n!-n}$ von $a^{m!-m}$ $(n \neq m)$ mittels a^n

Es gibt daher keine endlichen Automaten A mit $L_{fact}=L(A)$, $L_{ambm}=L(A)$, $L_{diff}=L(A)$.

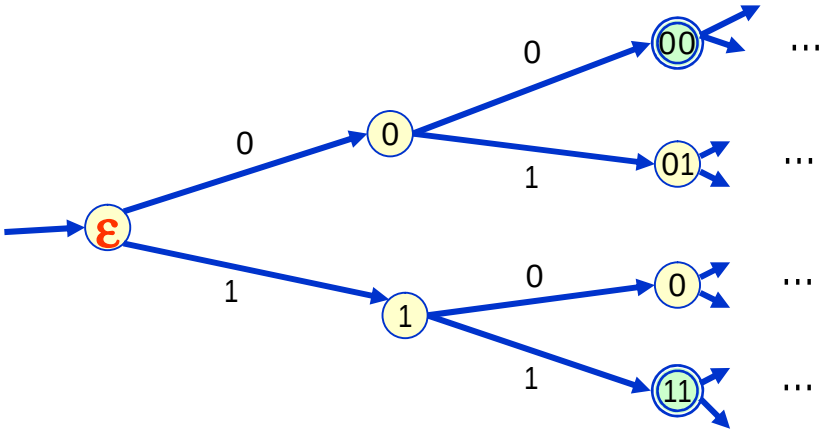
Unendliche Automaten

- Für jede Sprache $L \subseteq \Sigma^*$ gibt es einen unendlichen Automaten, der L akzeptiert

  - Beweis: $A = (\Sigma^*, \Sigma, \delta, \varepsilon, L)$, d.h.
    - Zustände : Worte über Σ
    - Anfangszustand: ε
    - Endzustände : die Worte aus L
    - $\delta(w,a) := wa$

  - Es folgt $\delta^*(w,v) = wv$ für alle $v,w \in \Sigma^*$ // Induktion

  - Dann gilt:
    - $w \in L(A)$ $\Leftrightarrow \delta^*(\varepsilon,w) \in L$ // Def. A
      $\Leftrightarrow \varepsilon w \in L$ // Eigenschaft δ^* (s.o.)
      $\Leftrightarrow w \in L$



unendlicher Automat für Palindrome

Minimalautomat

- Aus dem bisher gezeigten folgt
  - Ist jeder Zustand in A erreichbar, dann ist A/~ ein Automat mit minimaler Anzahl von Zuständen, der die gleiche Sprache erkennt
- Gegeben A, wie kann man A/~ konstruieren ?
  - Entferne alle nichterreichbaren Zustände
  - Berechne ~ und faktorisiere
    - schwer, wie soll man alle $w \in \Sigma^*$ durchprüfen ?
- Alternative:
  - Entferne alle nichterreichbaren Zustände
  - Wirf zunächst alles Zustände zusammen
  - trenne zwei Zustände p, q, falls nötig
    - Nötig heißt: trennbar
    - Geht effizient
    - muss maximal Worte der Länge |Q| betrachten

Axiome für trennbar

p,q trennbar $\Leftrightarrow \neg (p \sim q)$

$\sim$ war größte Relation mit

1. $p \sim q \Rightarrow (p \in F \wedge q \in F) \vee (p \notin F \wedge q \notin F)$
2. $\forall a \in \Sigma$:
   $p \sim q \Rightarrow \delta(p,a) \sim \delta(q,a)$

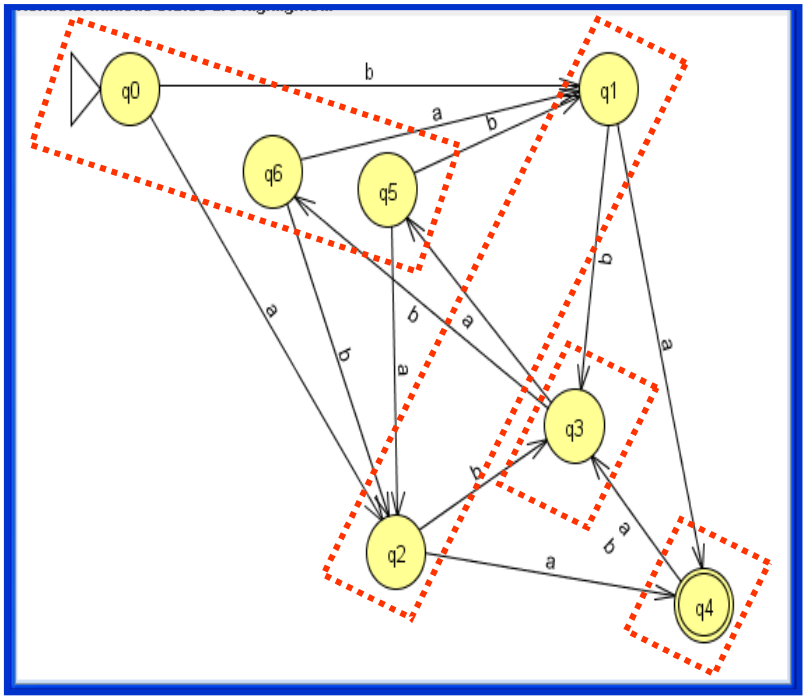
Mit der logischen Äquivalenz $(x \Rightarrow y) \Leftrightarrow (\neg y \Rightarrow \neg x)$ folgt:

trennbar ist kleinste Relation mit

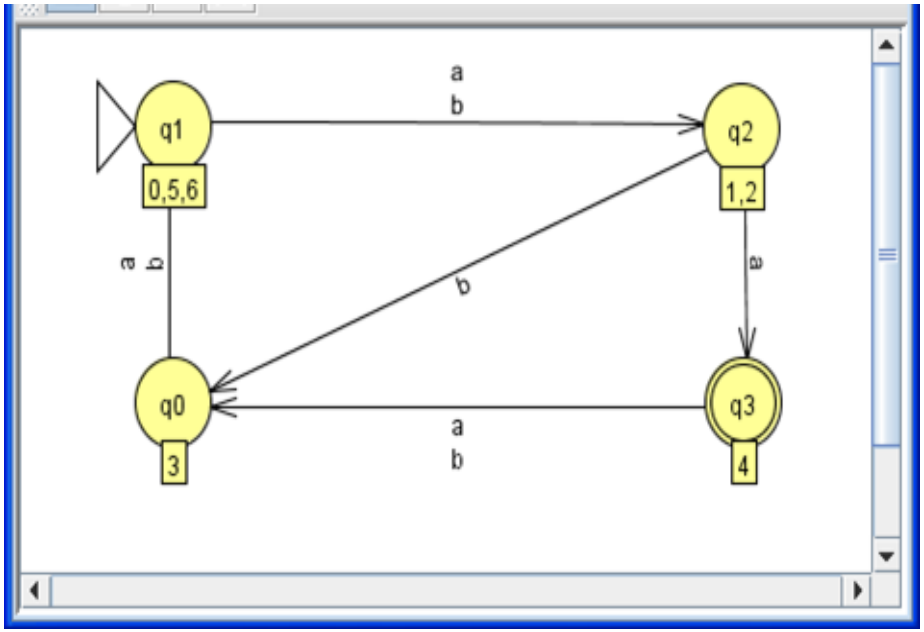
1. $(p \in F \wedge q \notin F) \Rightarrow p, q$ trennbar
2. $\forall a \in \Sigma$:
   $\delta(p,a), \delta(q,a)$ trennbar $\Rightarrow p,q$ trennbar

Minimierung

- A/~ ist Automat mit
  - L(A)=L(A/~)
  - Unter allen Automaten B mit L(A) = L(B) hat A/~ minimale Anzahl von Zuständen
- Wie kann man A/~ effizient konstruieren ?



A



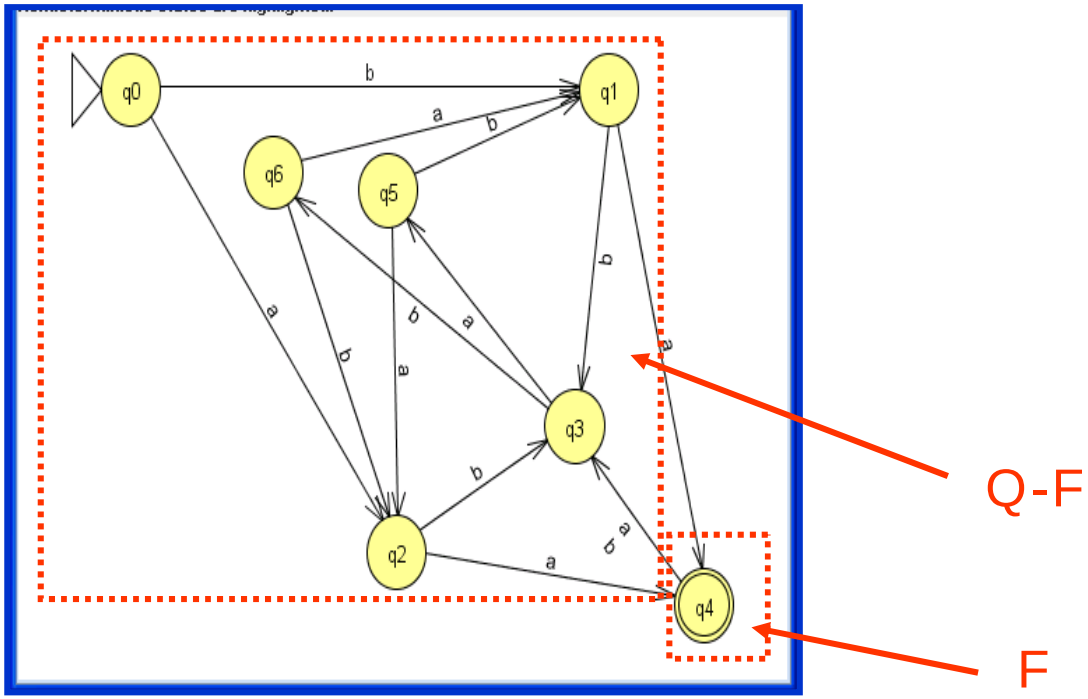
A/~

Minimierung

- A/~ ist Automat mit
  - L(A)=A(A/~)
  - Unter allen Automaten B mit L(A) = L(B) hat A/~ minimale Anzahl von Zuständen
- Wie kann man A/~ effizient konstruieren ?

Maxime: trenne Zustände von A nur wenn notwendig

1. Trenne Endzuständen von Nicht-Endzuständen



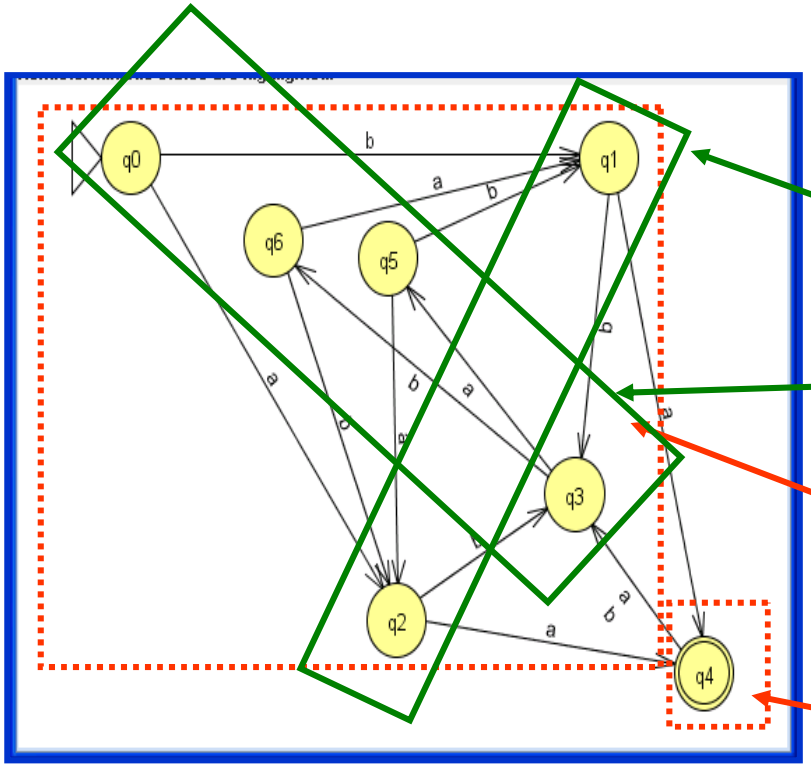
A

Minimierung

- A/~ ist Automat mit
  - L(A)=A(A/~)
  - Unter allen Automaten B mit L(A) = L(B) hat A/~ minimale Anzahl von Zuständen
- Wie kann man A/~ effizient konstruieren ?

Maxime: trenne Zustände von A nur wenn notwendig

1. Trenne Endzuständen von Nicht-Endzuständen
2. Wähle Zustände p,q, die noch nicht getrennt sind und $a \in \Sigma$. Wenn $\delta(p,a)$, $\delta(q,a)$ schon getrennt, dann trenne auch p von q.



$\{ p \in Q \mid \delta(p,a) \in F \}$

$\{ p \in Q \mid \delta(p,a) \in Q-F \}$

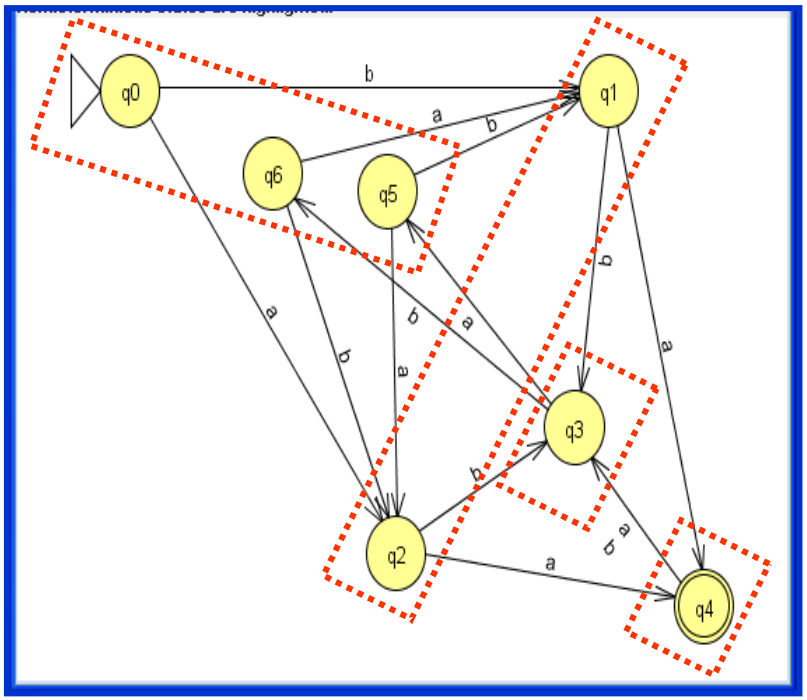
Q-F

F

A

Minimierung

- A/~ ist Automat mit
  - L(A)=A(A/~)
  - Unter allen Automaten B mit L(A) = L(B) hat A/~ minimale Anzahl von Zuständen
- Wie kann man A/~ effizient konstruieren ?

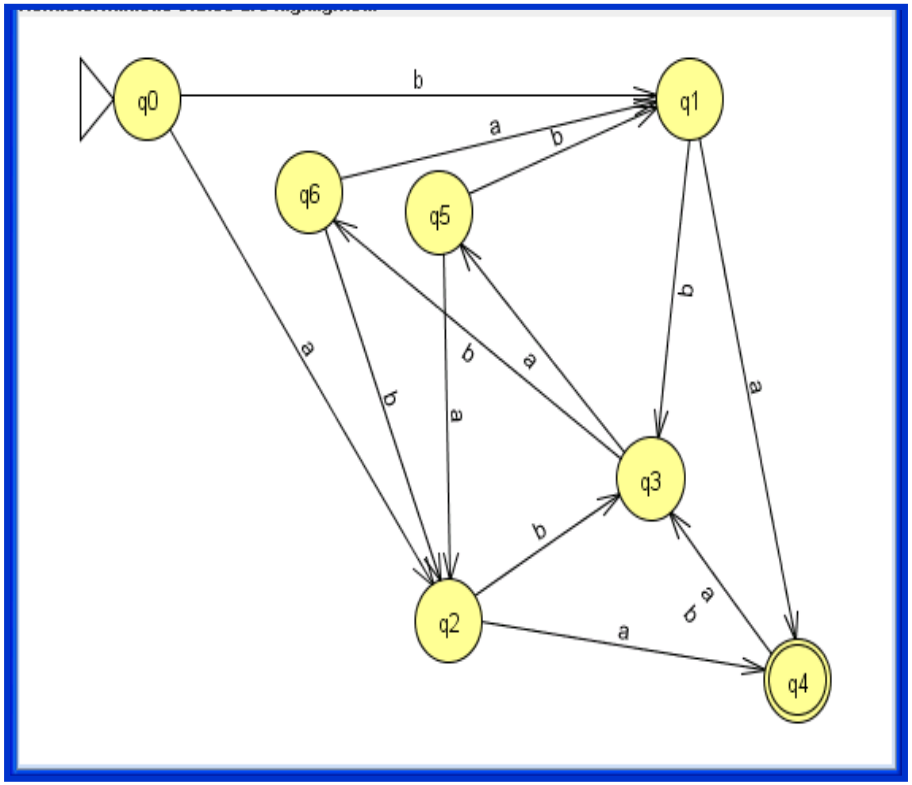


A

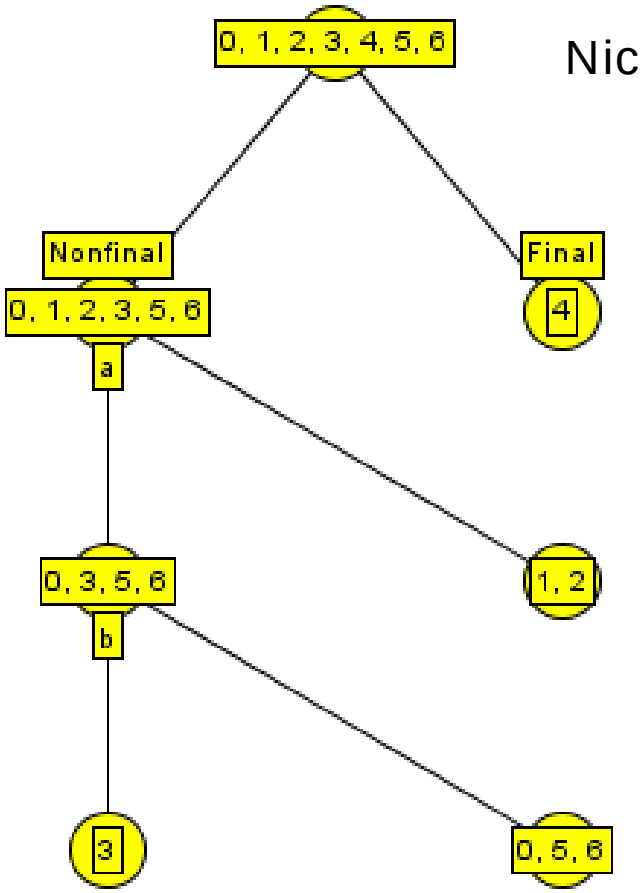
Maxime: trenne Zustände von A nur wenn notwendig

1. Trenne Endzuständen von Nicht-Endzuständen
2. Wähle Zustände p,q, die noch nicht getrennt sind und $a \in \Sigma$. Wenn $\delta(p,a)$, $\delta(q,a)$ schon getrennt, dann trenne auch p von q.
3. Bis nichts mehr getrennt wird

Darstellung in JFLAP



Partitionsbaum



Nichts getrennt

Q-F von F

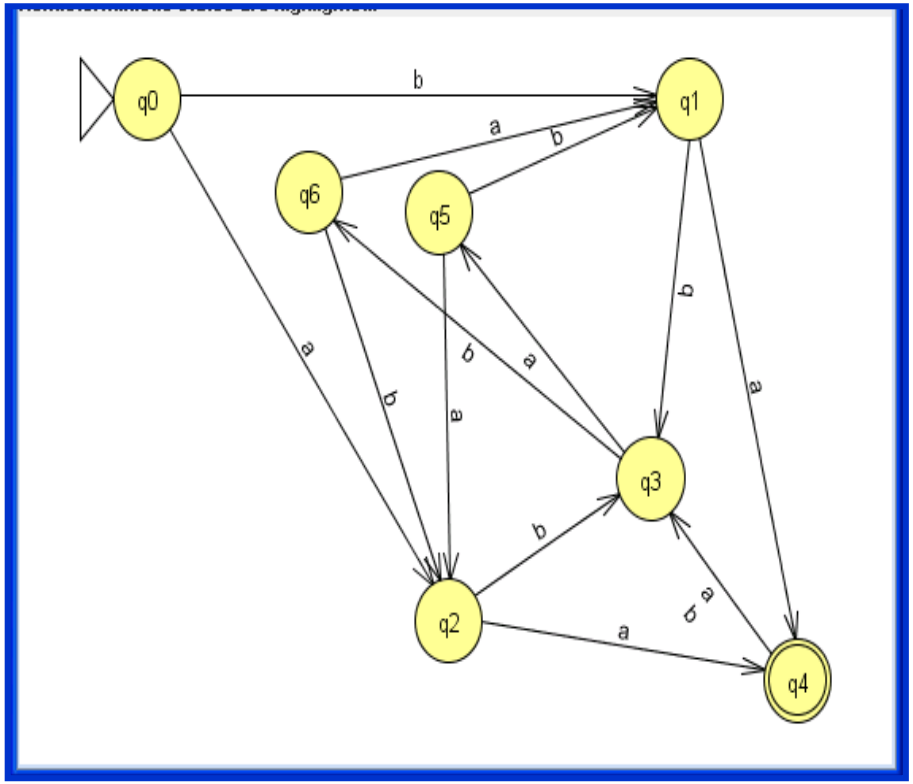
Trennung nach a

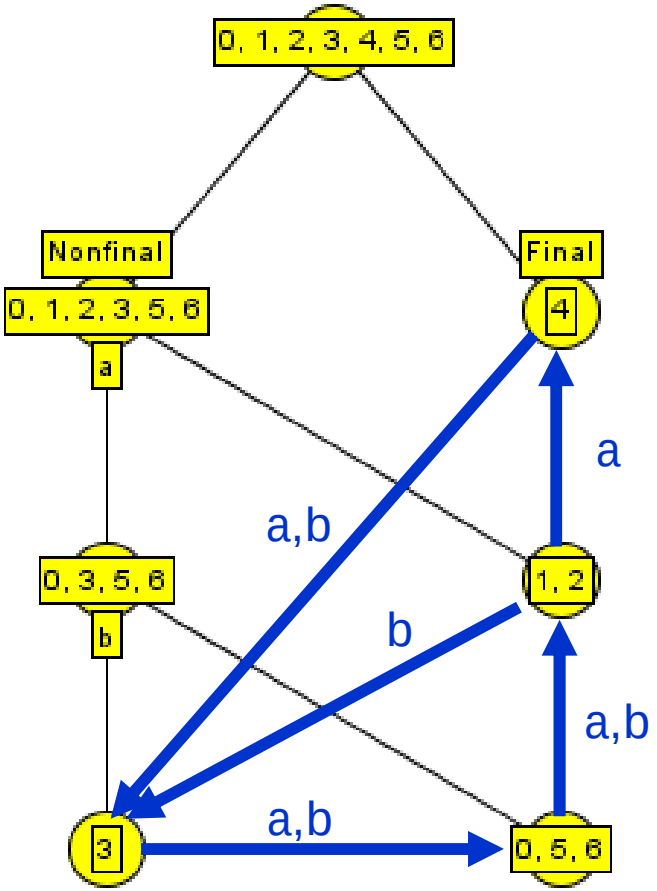
Trennung nach b

Schon fertig (Kleines Beispiel)

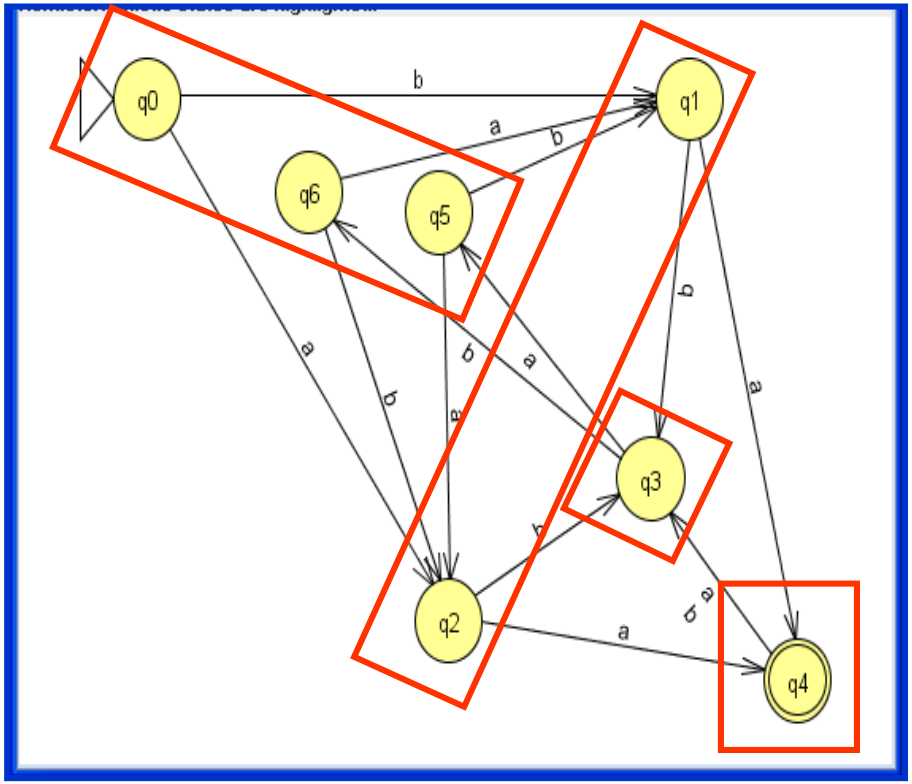
Fertiger Automat

- Die Zustände des Minimalautomaten sind die Blätter des Partitionsbaumes
- Transitionen: Repräsentantenweise



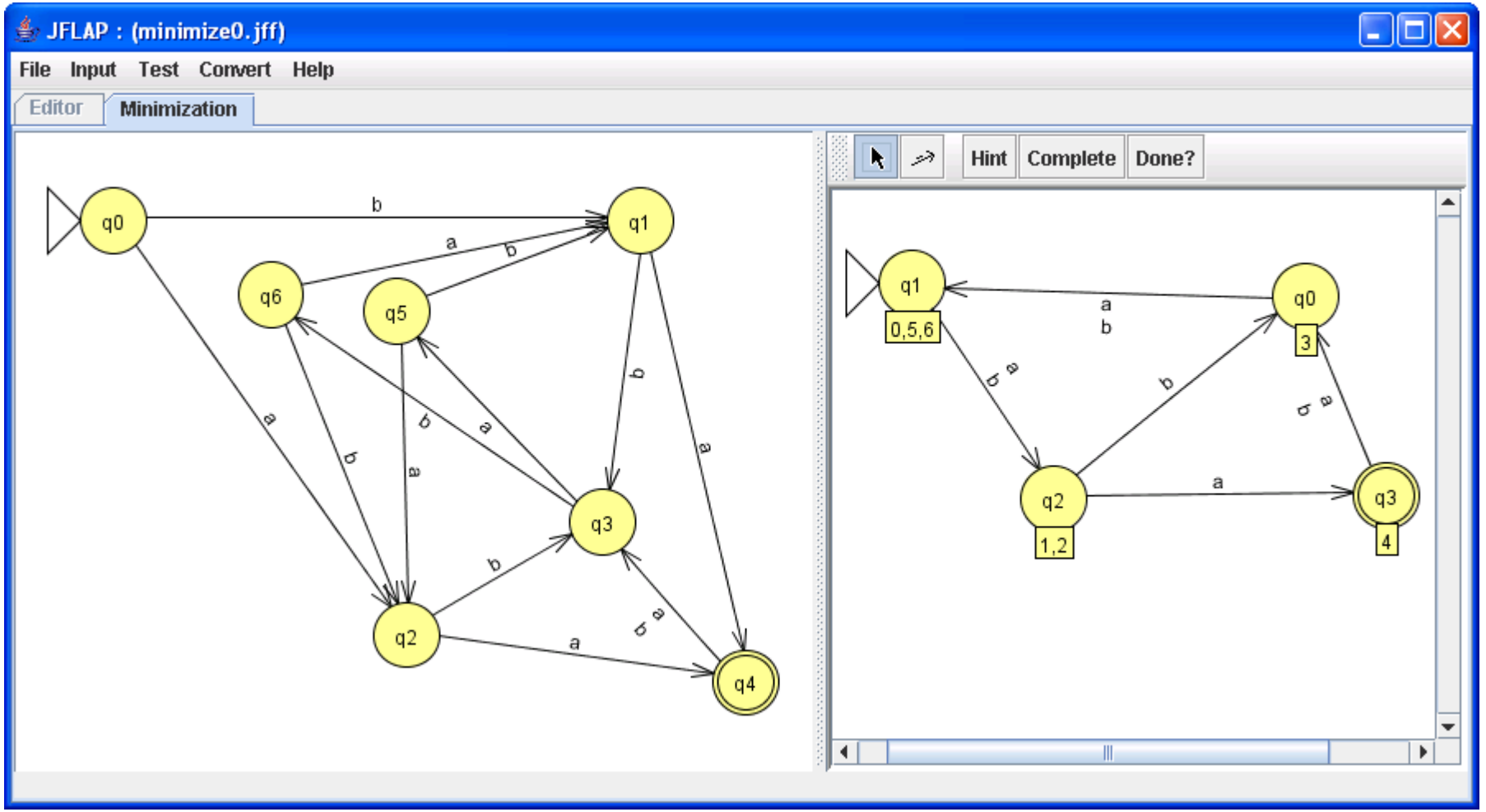


Andere Möglichkeit: Tabelle



- Tabelliere Trennbarkeitsrelation
  - Trenne F und Q-F
  - Für alle e $\in \Sigma$,
    für alle p < q
    Falls $\delta(p,a)$, $\delta(q,a)$ schon getrennt,
    dann trenne p und q
  - Wiederhole bis eine Runde lang nichts getrennt wurde

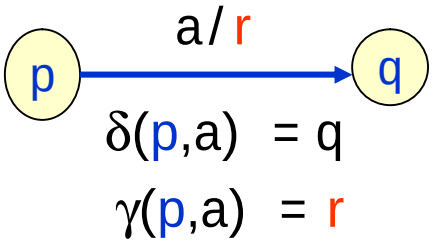
	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_0		x	x	x	x		
q_1				x	x	x	x
q_2				x	x	x	x
q_3					x	x	x
q_4						x	x
q_5							
q_6							



In JFLAP

Automaten mit Ausgabe

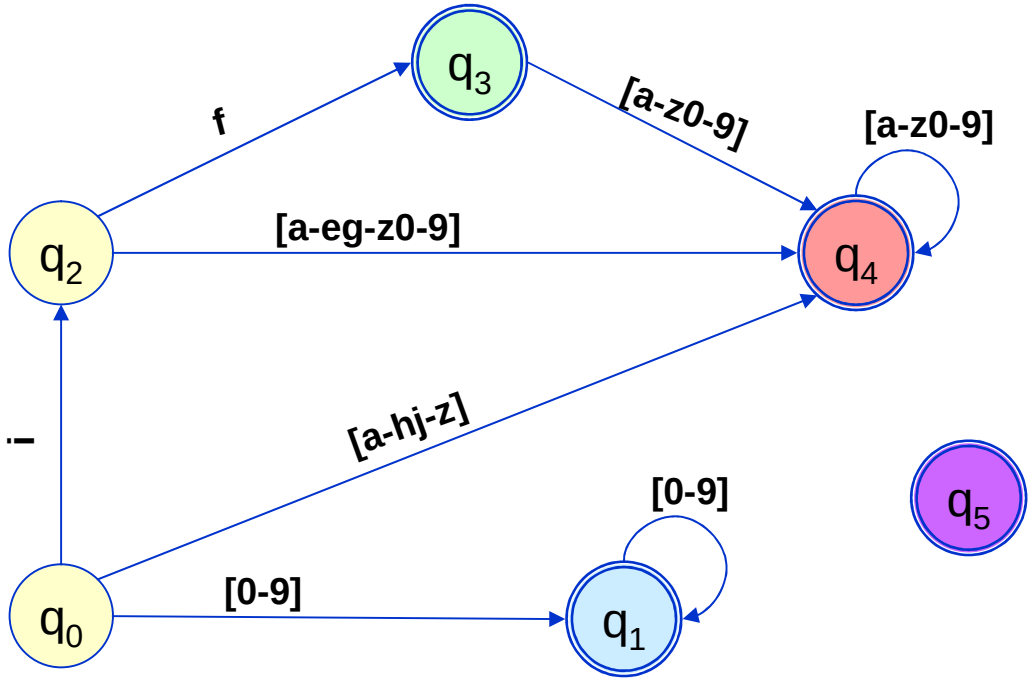
- Automaten mit Ausgabe haben zusätzlich
  - ein Ausgabealphabet Γ
  - eine Ausgabefunktion
    - entweder $\gamma : Q \rightarrow \Gamma$ (Moore-Automat)
    - oder $\gamma : Q \times \Sigma \rightarrow \Gamma$ (Mealy-Automat)



- Automaten ohne Ausgabe heißen demgegenüber auch: Akzeptoren
- Automaten mit Ausgabe: Transducer

Scanner als Automat

- Scanner: Automaten, die mehrere Sprachen erkennen
  - Jede Sprache entspricht Teilmenge der Endzustände
  - Ausgabe-Token hält fest, aus welcher Sprache das erkannte Wort ist
  - Versuche, möglichst langes Präfix zu erkennen
  - dann wird abgeschnitten



$q_3 \Rightarrow$ *ifToken*
$q_4 \Rightarrow$ *identifier*
$q_1 \Rightarrow$ *num*

fehlende Transitionen gehen auf
$q_5 \Rightarrow$ *Error*