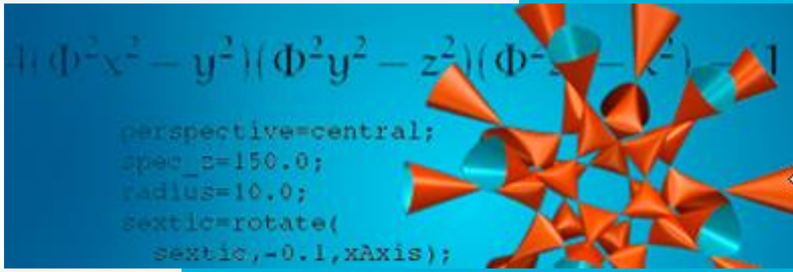
Theoretische Informatik: Deterministische Automaten

Prof. Dr. Elmar Tischhauser



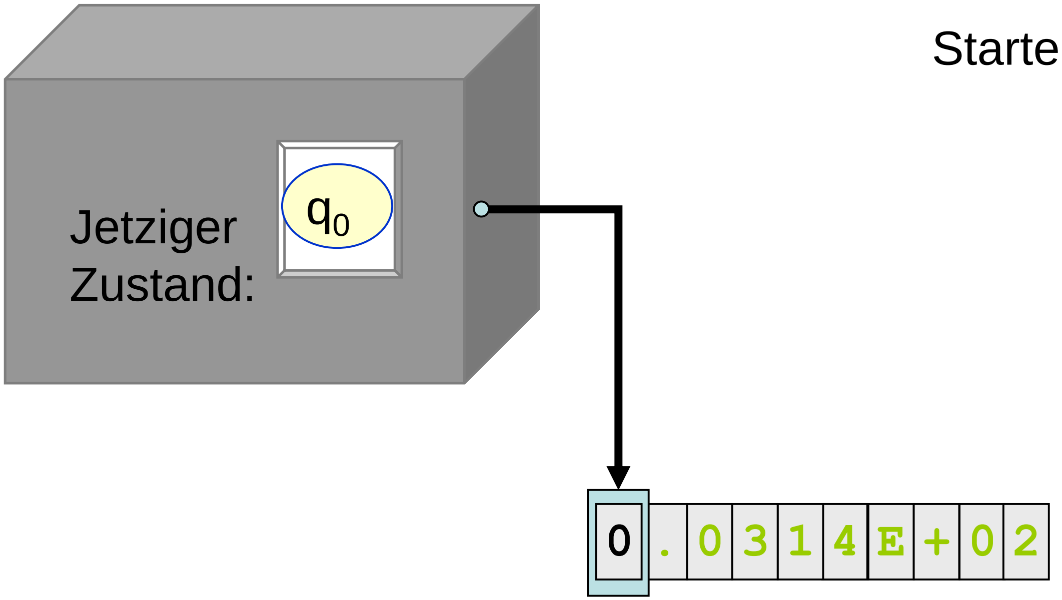
Inhalt

1. Deterministische Akzeptoren
   - Sprache eines Automaten
   - Implementierung
   - Komplement, Produkte
   - Faktorautomat
   - Minimalautomat
2. Grenzen von Automaten
   - Trennbarkeit
   - Nerode-Lemma
   - Pumping Lemma
   - Automaten mit Ausgabe

Automaten

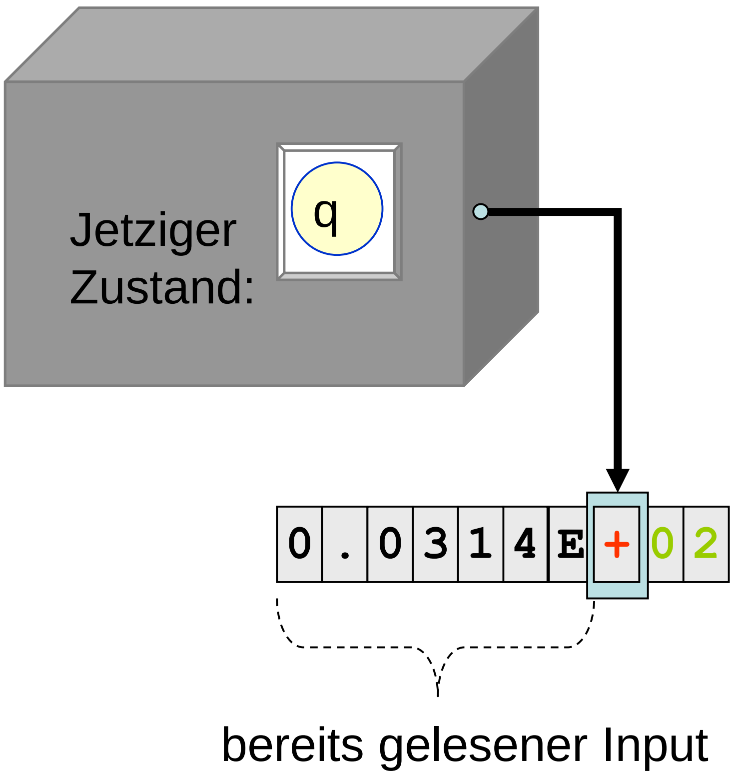
- sollen Sprache erkennen

Starten in Anfangszustand q_0



Automaten

- sollen Sprache erkennen

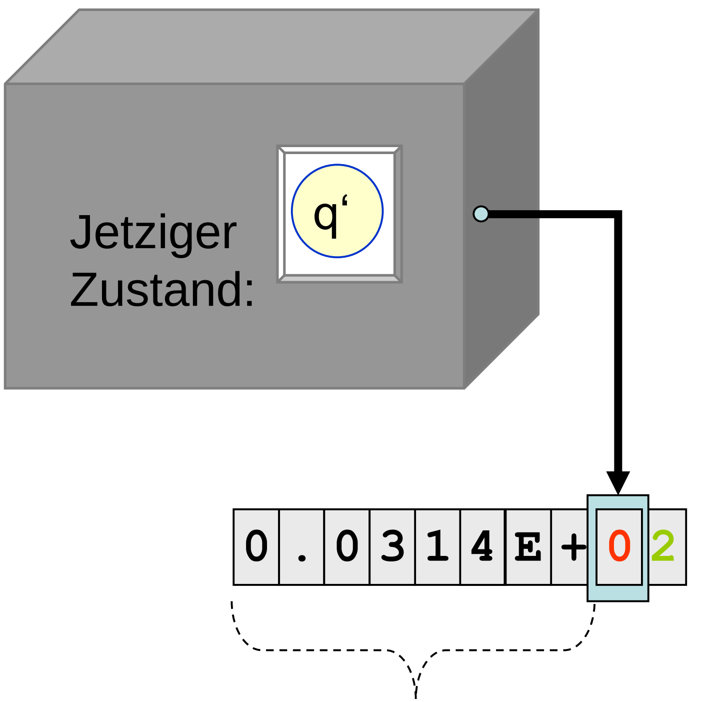


Starten in Anfangszustand q_0

abhängig von
gegenwärtigem Zustand q
gelesenem Zeichen a
neuer Zustand $q' = \delta(q,a)$

Automaten

- sollen Sprache erkennen



bereits gelesener Input

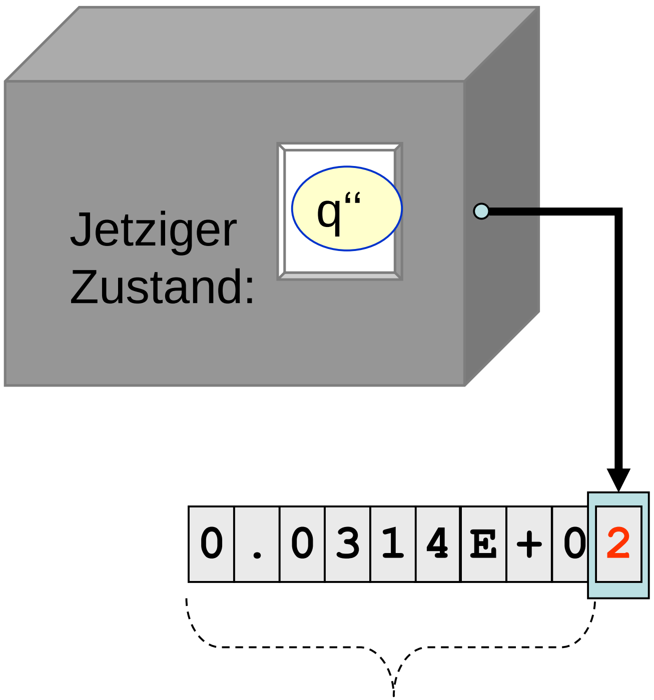
Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

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

Automaten

- sollen Sprache erkennen



bereits gelesener Input

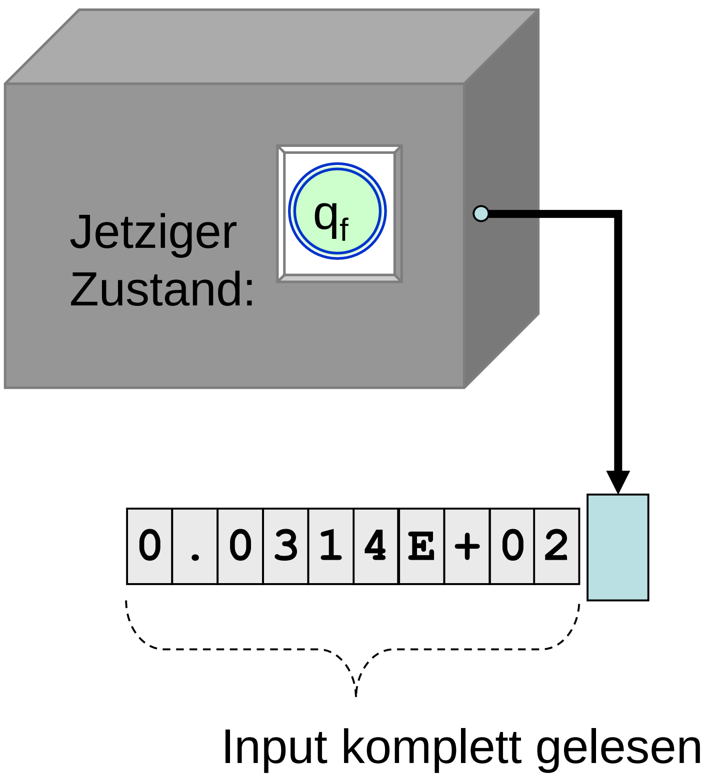
Starten in Anfangszustand q_0

abhängig von
- gegenwärtigem Zustand q
- gelesenem Zeichen a

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

Automaten

- sollen Sprache erkennen



Starten in Anfangszustand q_0

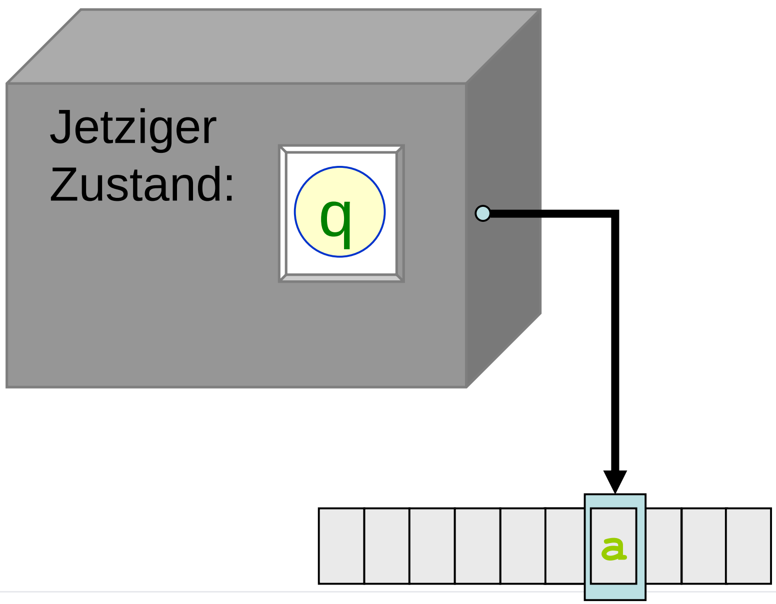
abhängig von
gegenwärtigem Zustand q
gelesenem Zeichen a
neuer Zustand $q' = \delta(q,a)$

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



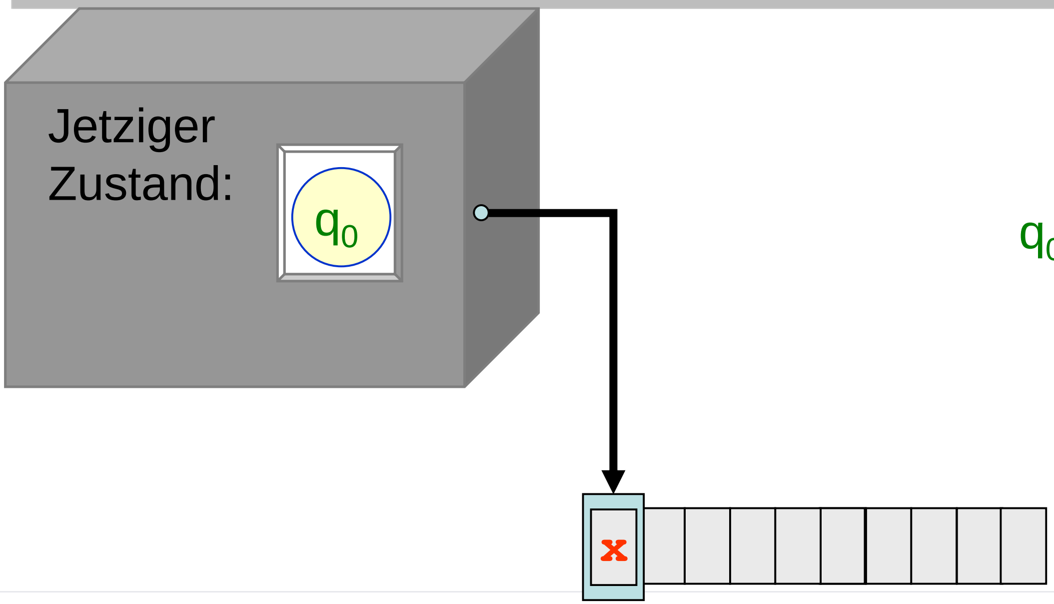
$q \in Q$

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

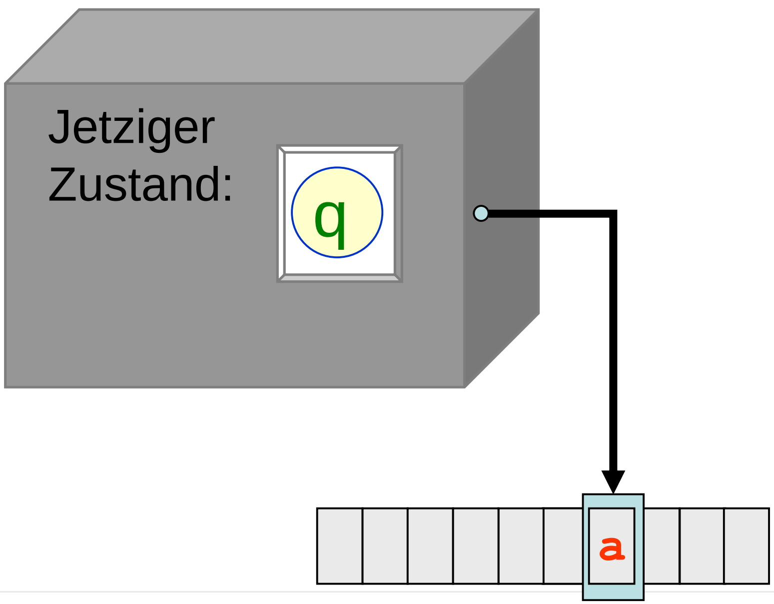


$q_0 \in Q$

Definition: Automat (DFA)

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

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



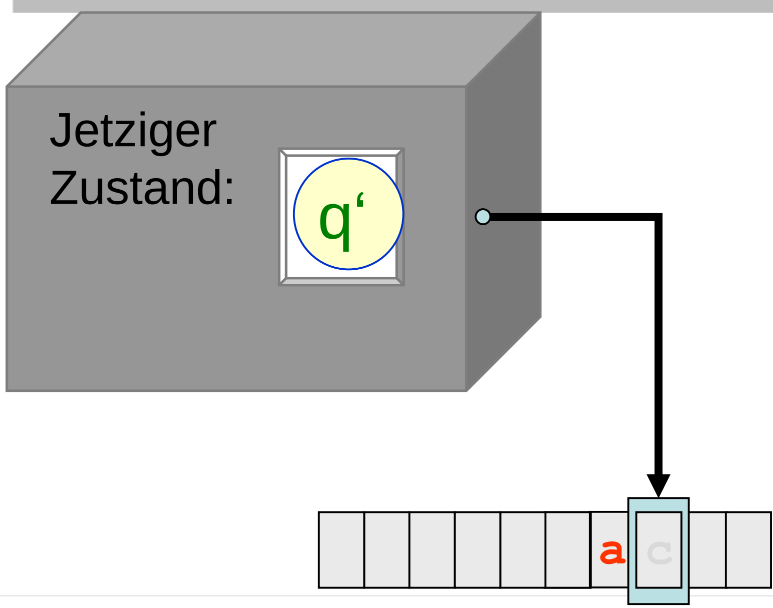
neuer Zustand

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

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



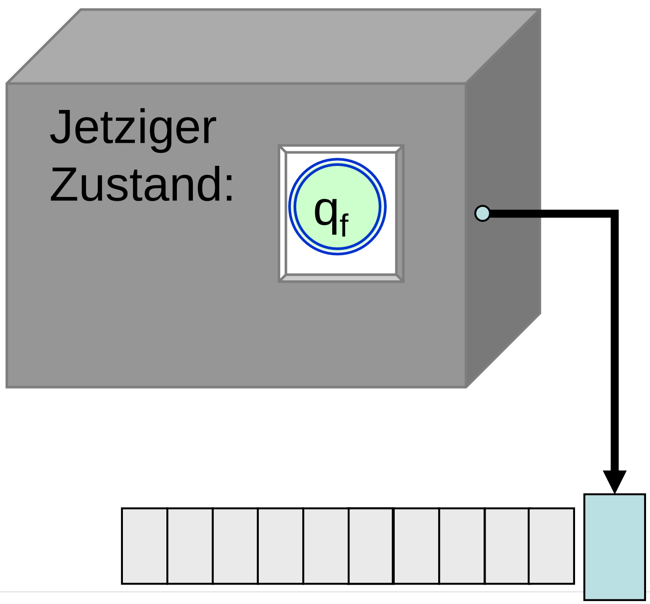
neuer Zustand

$\delta(q,a)=q‘$

Definition: Automat (DFA)

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

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



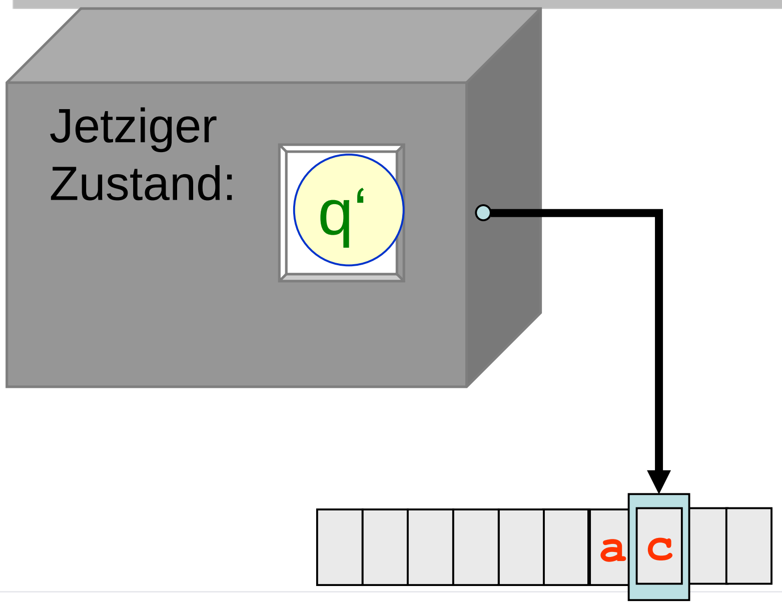
$q_f \in F \subseteq Q$

Definition: Automat (DFA)

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

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

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

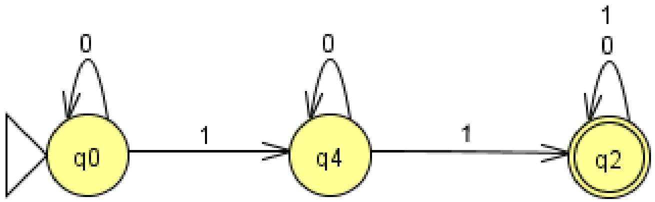


neuer Zustand

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

Zustandsübergangsgraphen

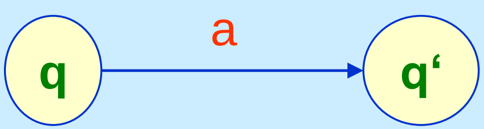
- Graphische Notation für Automaten
  - Zustände - Knoten
  - ein Anfangszustand $q_0 \in Q$
  - Zustandsübergänge – beschriftete Kanten
  - eine Menge von Endzuständen $F \subseteq Q$
- Wort wird akzeptiert, falls es
  - beginnend im Anfangszustand
  - den Automaten in einen Endzustand überführt



zweiEinsen.jff

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

$\Leftrightarrow$



Im Beispiel:
$Q = \{q_0, q_4, q_2\}$
$F = \{q_2\}$
$\delta(q_0,0) = q_0$
$\delta(q_0,1) = q_4$, etc

1101 akzeptiert
0010 nicht akzeptiert

Beispiel: BinInteger

- Automat über dem Alphabet $\Sigma=\{0,1,+,-\}$.

- Erkennt alle gültigen Binären Integer
  - optionales Vorzeichen
  - keine führenden 0-en erlaubt

- Entspricht dem regulären Ausdruck
  - (- | + | ε) (0 | 1 (0 | 1)*)

    = (- | +)? (0 | 1 (0 | 1)*)

q_4 „*Error“-Zustand*

- nicht akzeptierend
- kann nicht verlassen werden

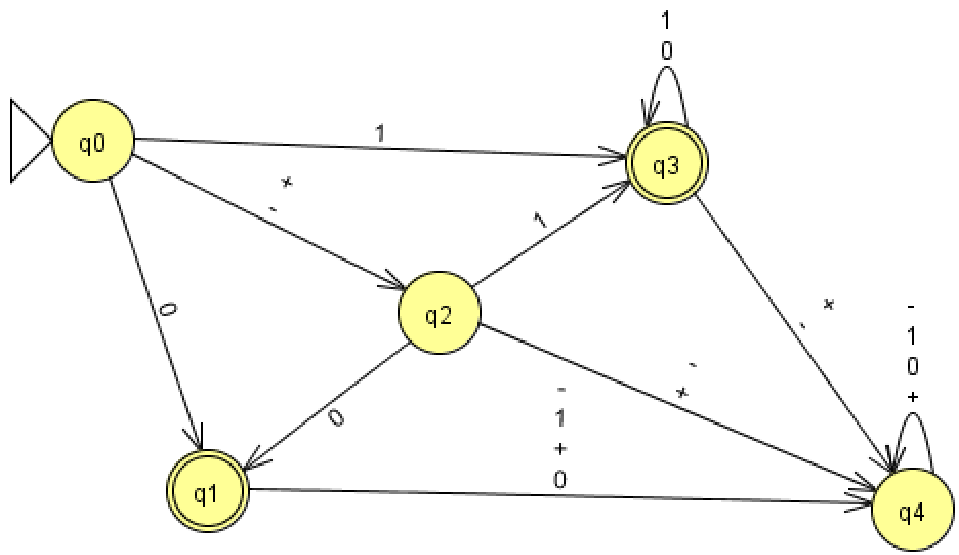
<u>Automat</u>:

$Q := \{ q_0, q_1, q_2, q_3, q_4 \}$

q_0 : Anfangszustand

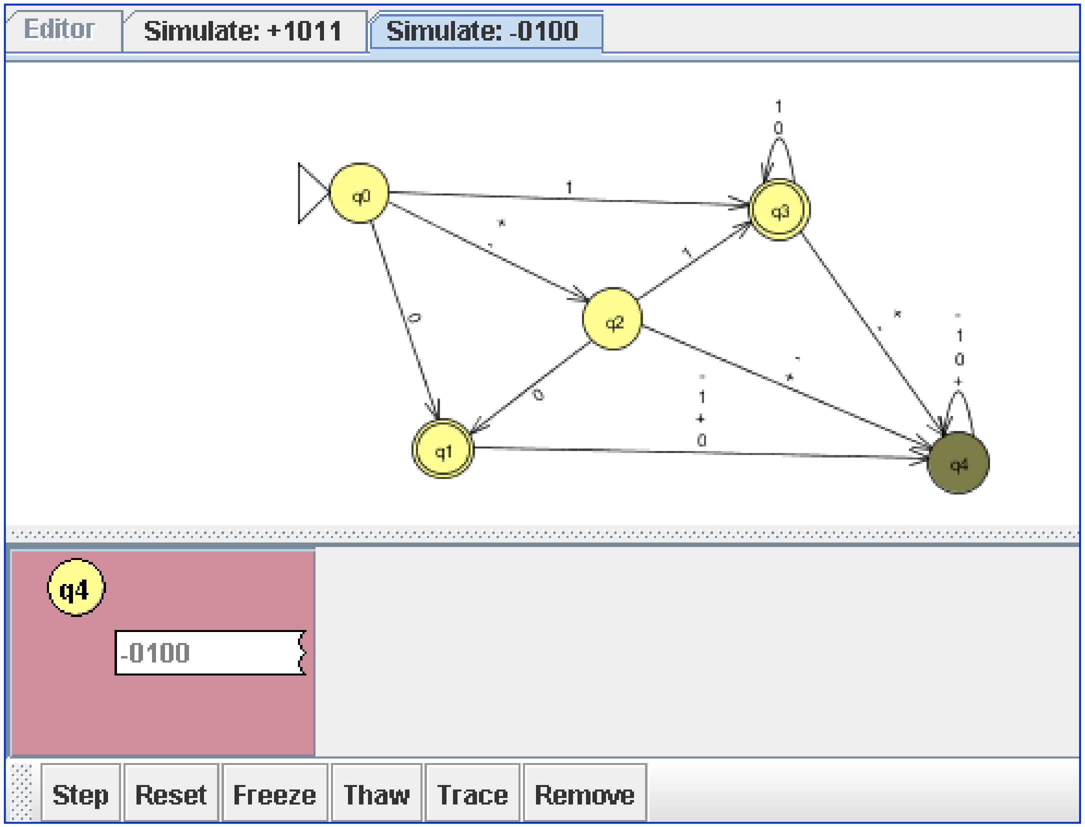
$F := \{ q_1, q_3 \}$ Endzustände

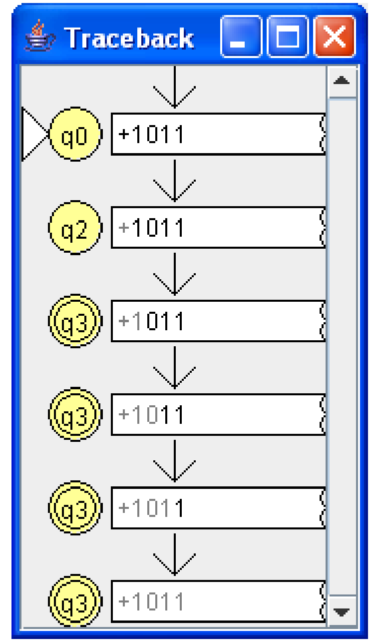
δ : durch folgenden Übergangsgraphen definiert:



Akzeptanz

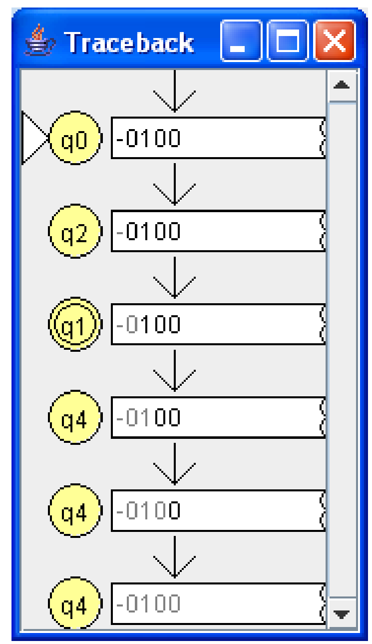
- Ein Wort $w \in \Sigma^*$ wird akzeptiert, falls es vom Anfangszustand in einen Endzustand führt





JFLAP output

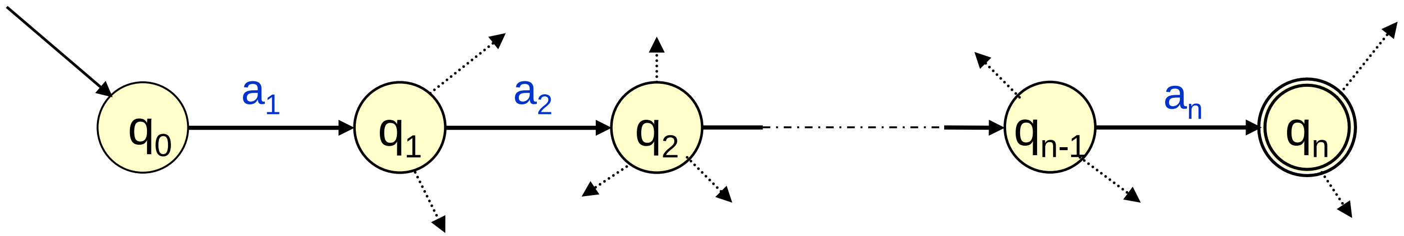
+1011 führt zu $q_3 \in F$ $\Rightarrow$ akzeptiert



-0100 führt zu $q_4 \notin F$ $\Rightarrow$ nicht akzeptiert

Läufe

- Wort beschreibt Weg durch den Automaten
  - Wort $w=a_1a_2\ldots a_n$ 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



Lauf für das Wort $w = a_1a_2\ldots a_n$

Hier: $\delta(\delta(\ldots\delta(\delta(q_0, a_1), a_2)\ldots), a_n) = q_n$

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

δ^*

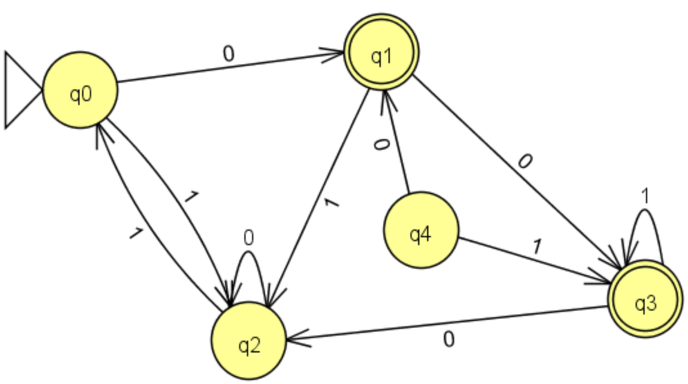
- Ausdehnung von δ auf Worte

- $\delta^* : Q \times \Sigma^* \rightarrow Q$ definiert durch

  – $\delta^*(q, \boldsymbol{\varepsilon}) := q$ // 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 $\delta^*(q_0, w) = q$



Beispiele

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

q_4 ist nicht erreichbar

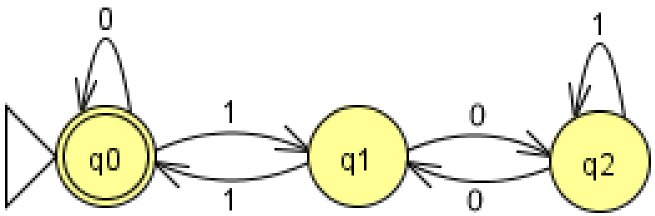
Sprache eines Automaten

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 :

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

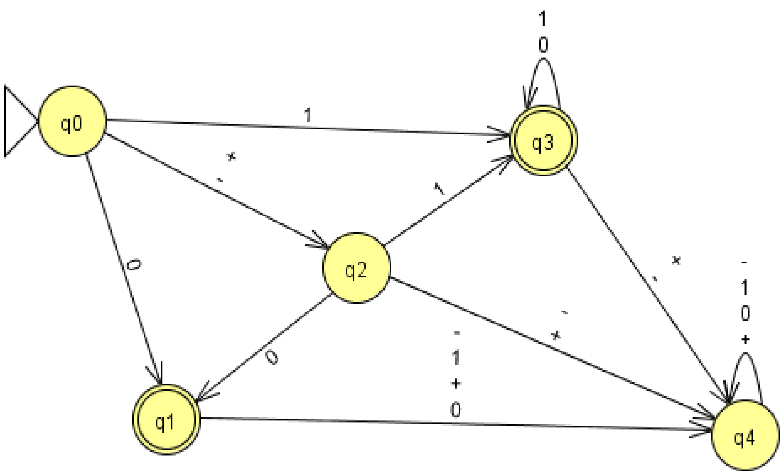


<u>Beispiel</u>: $w \in L(A) \Leftrightarrow$
$w = \varepsilon$ oder $(w)_2 \bmod 3 = 0$

Implementierung

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

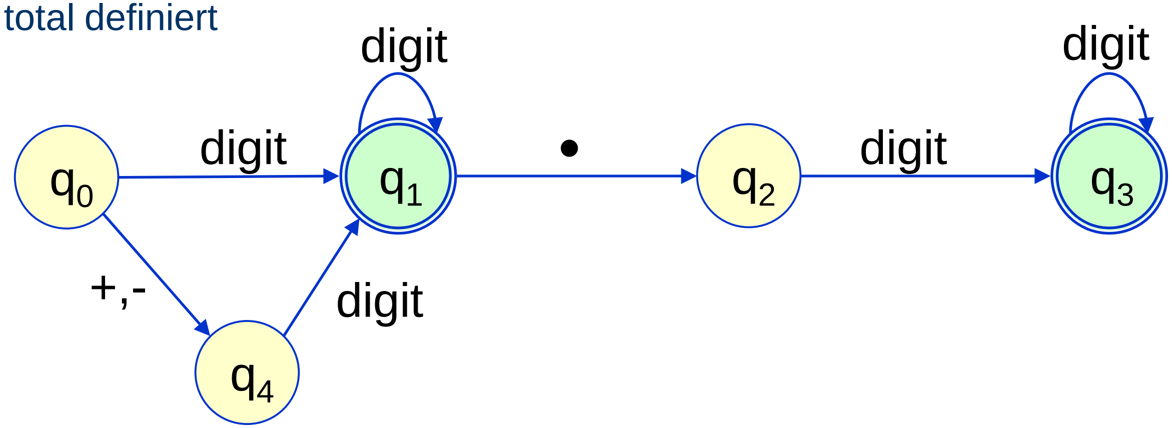
$F = \{ q_1, q_3 \}$
init = q_0.



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

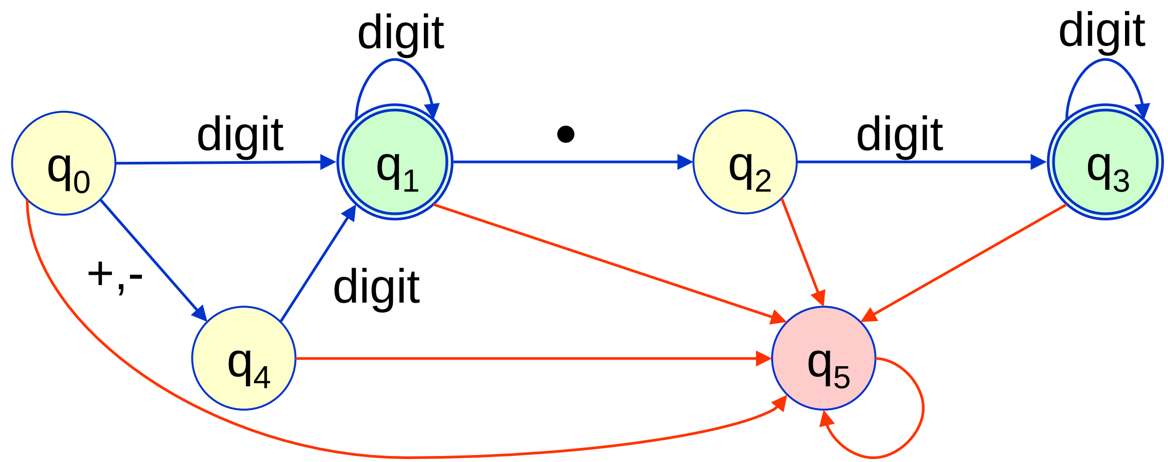
- Noch kein DFA
  - δ nicht total definiert



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



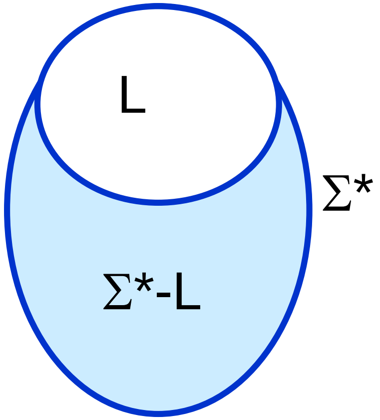
δ	+	-	Ziffer	.
q_0	q_4	q_4	q_1	q_5
q_1	q_5	q_5	q_1	q_2
q_2	q_5	q_5	q_3	q_5
q_3	q_5	q_5	q_3	q_5
q_4	q_5	q_5	q_1	q_5
q_5	q_5	q_5	q_5	q_5

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

Komplementautomat

- Satz: 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:

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

$\iff \delta^*(q_0,w) \notin F$

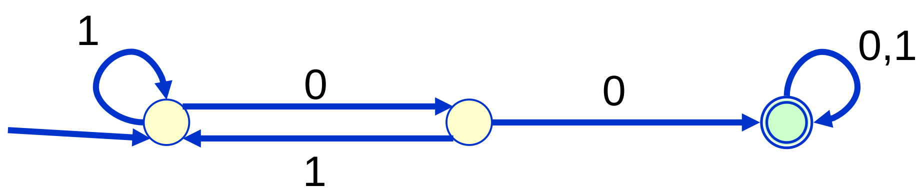
$\iff w \notin L(A)$

Produktautomat

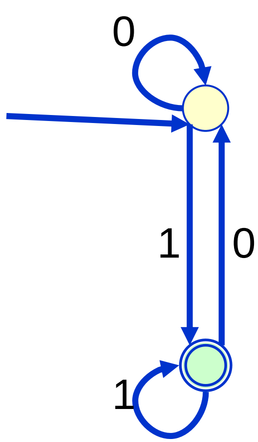
$A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$ seien Automaten

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



A



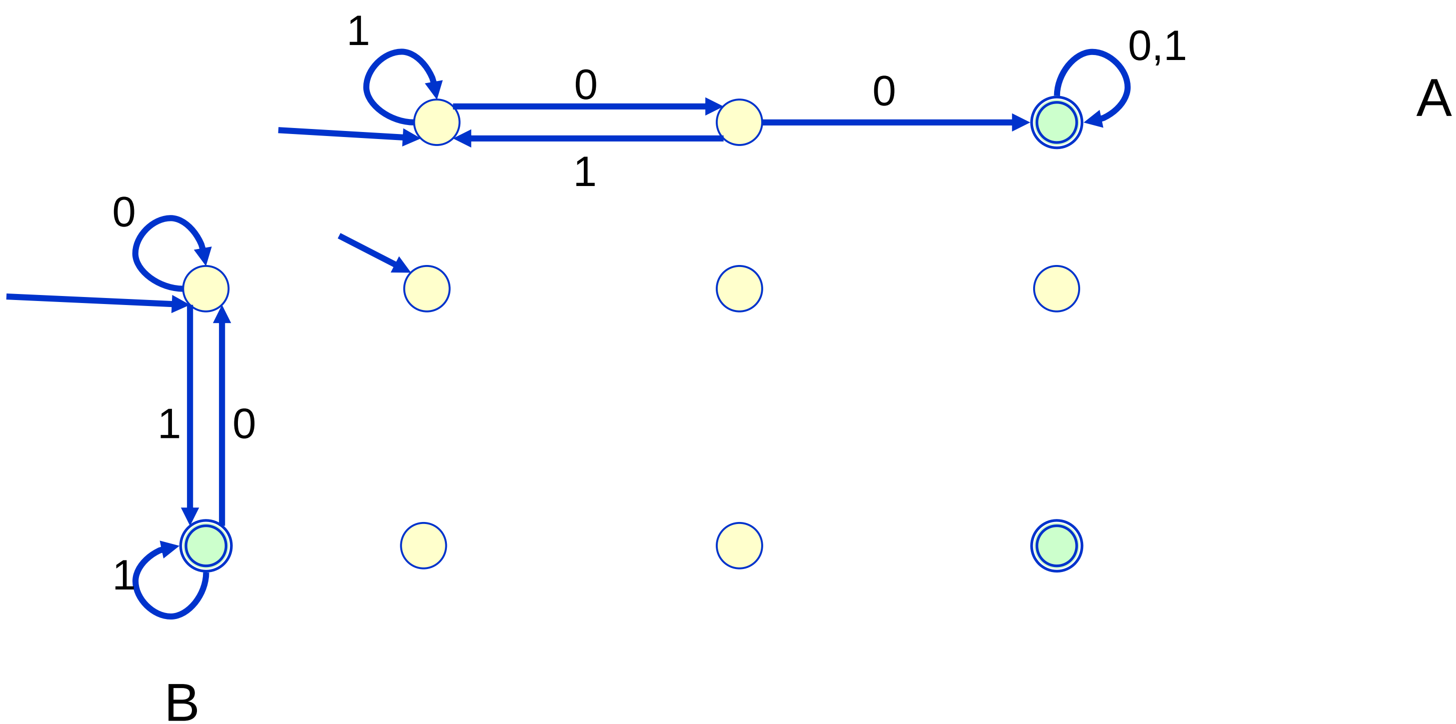
B

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

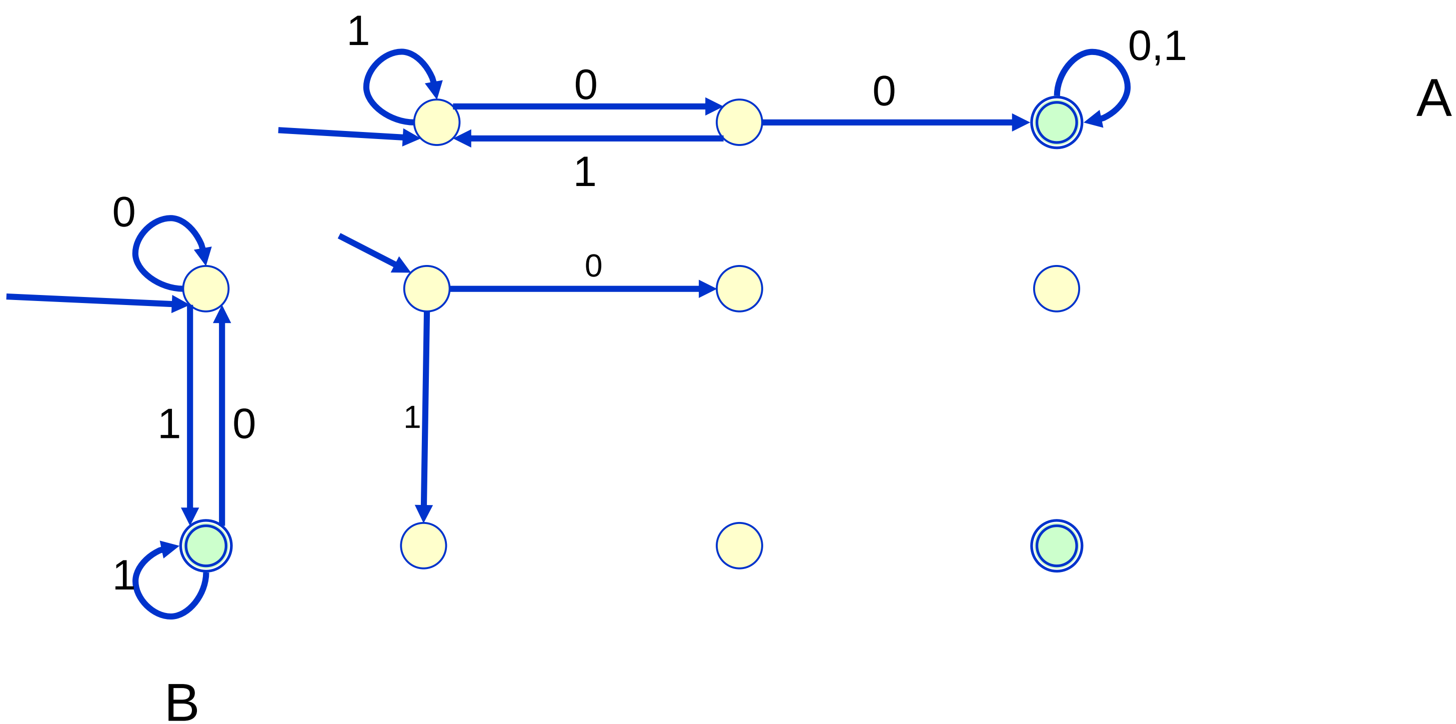


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

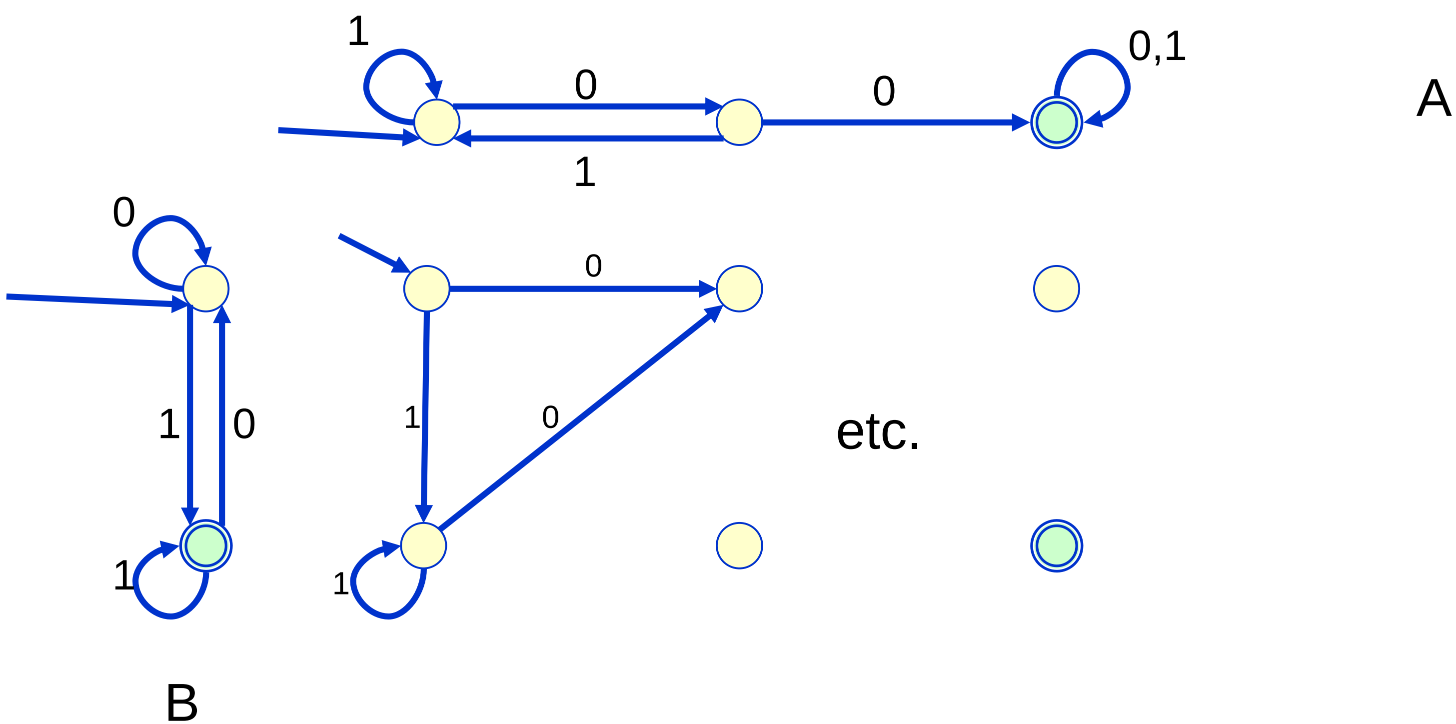


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

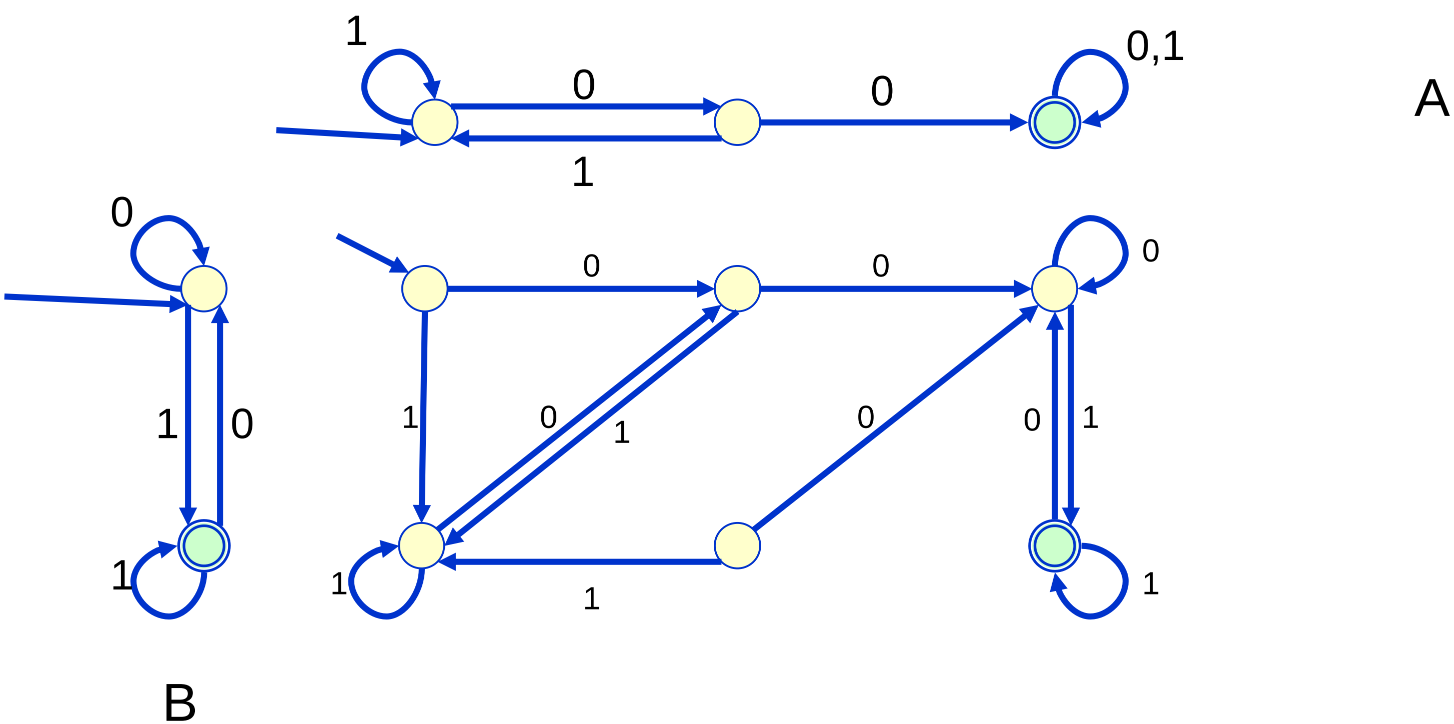


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

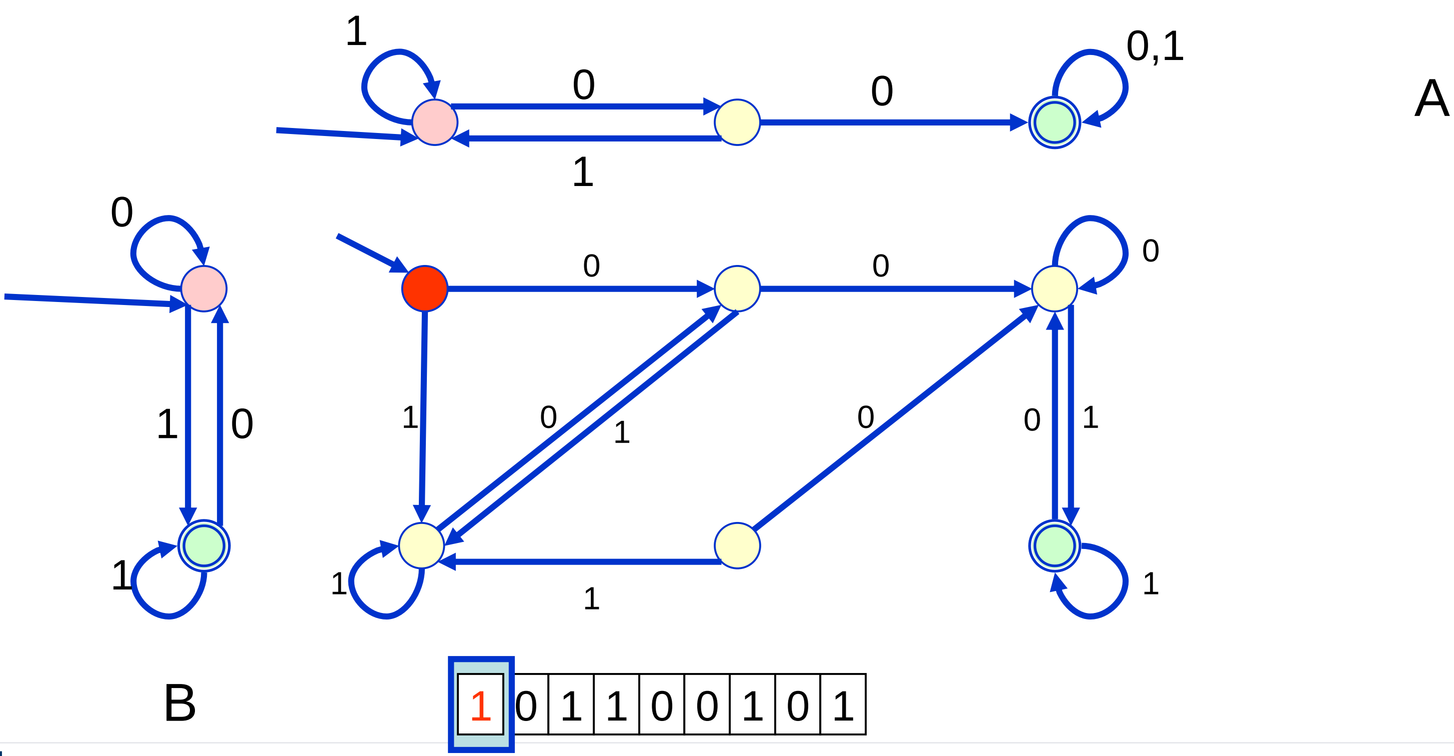


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

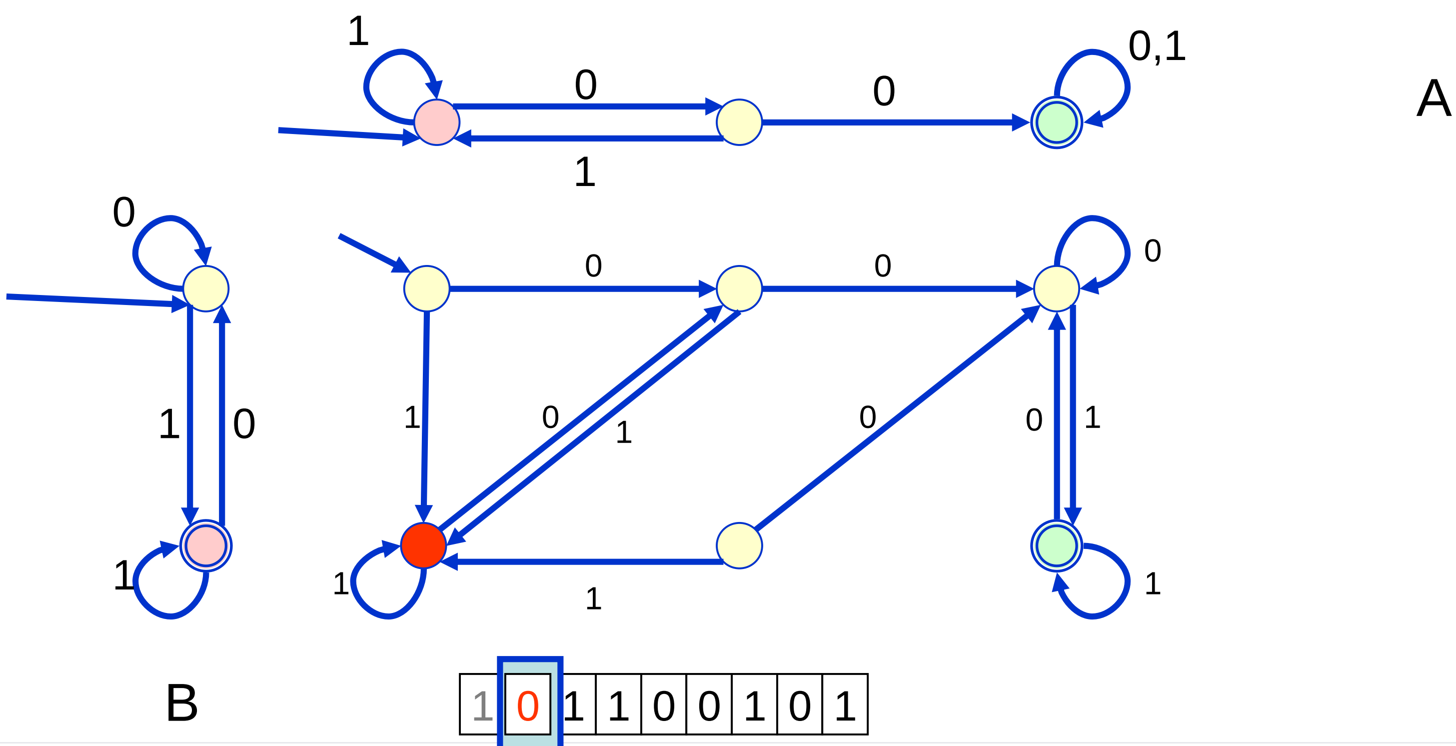


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

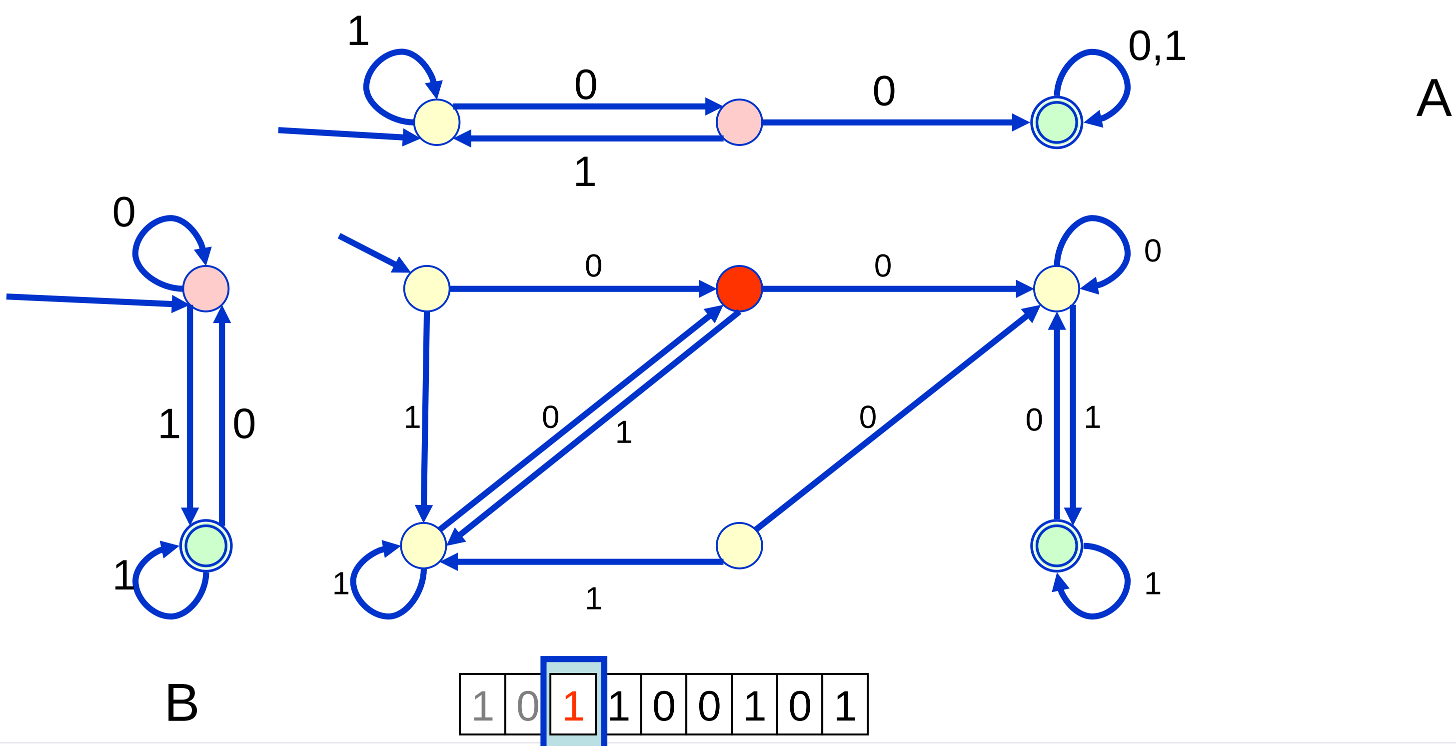


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

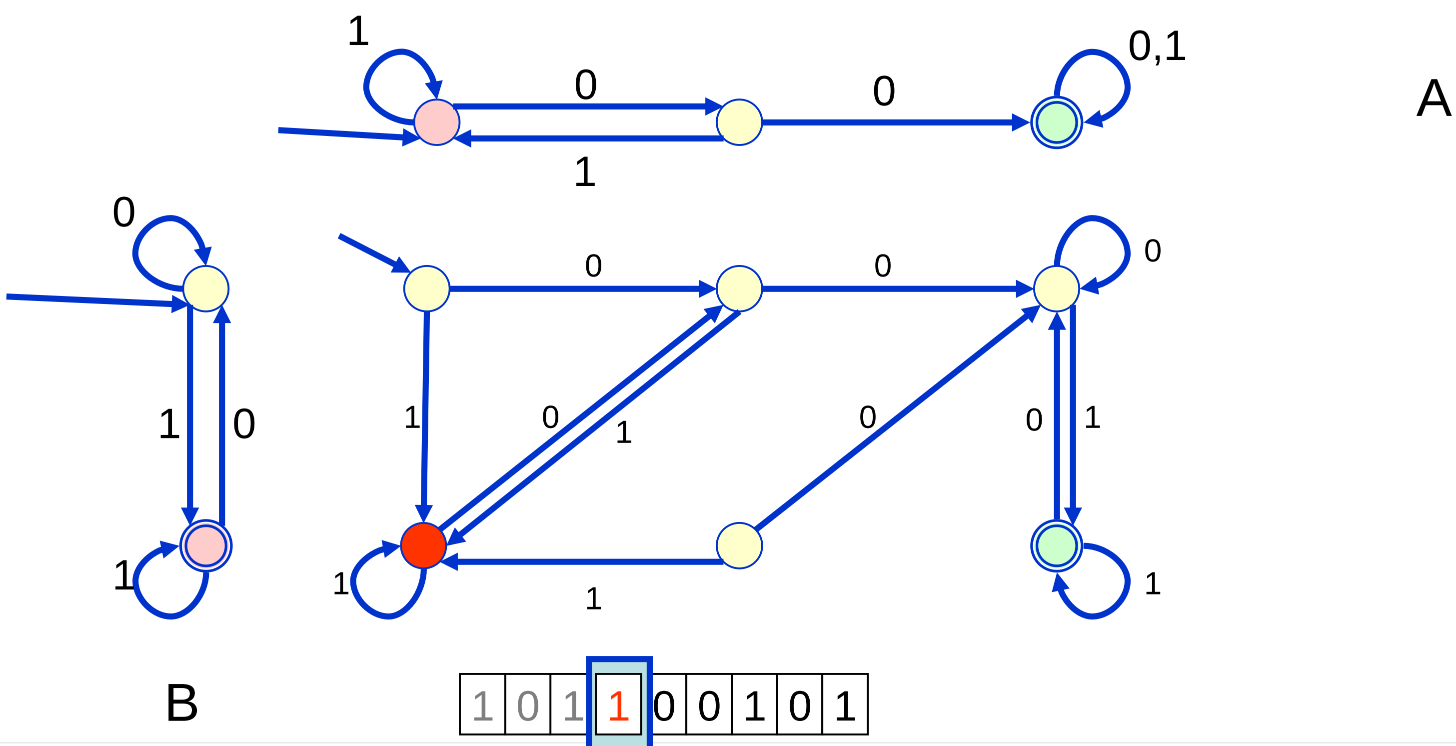


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

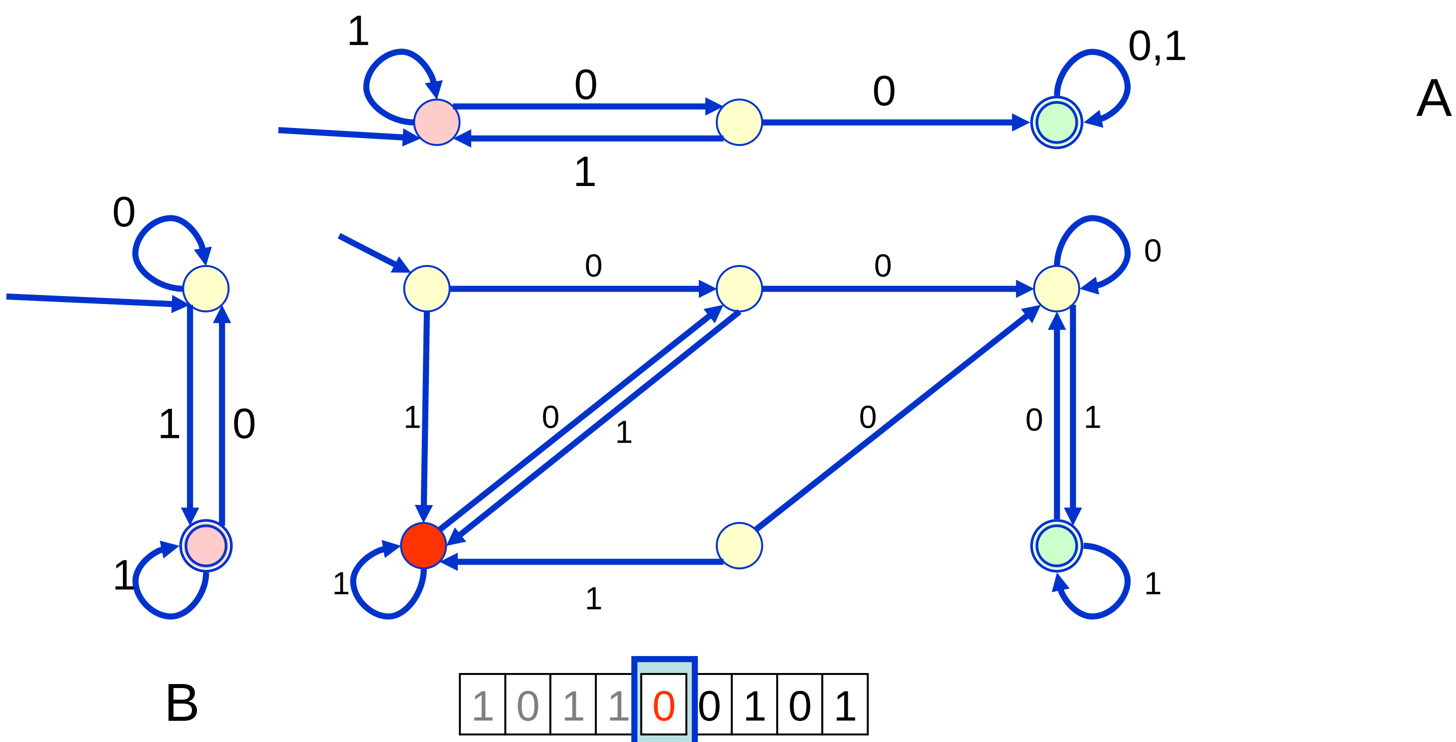


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

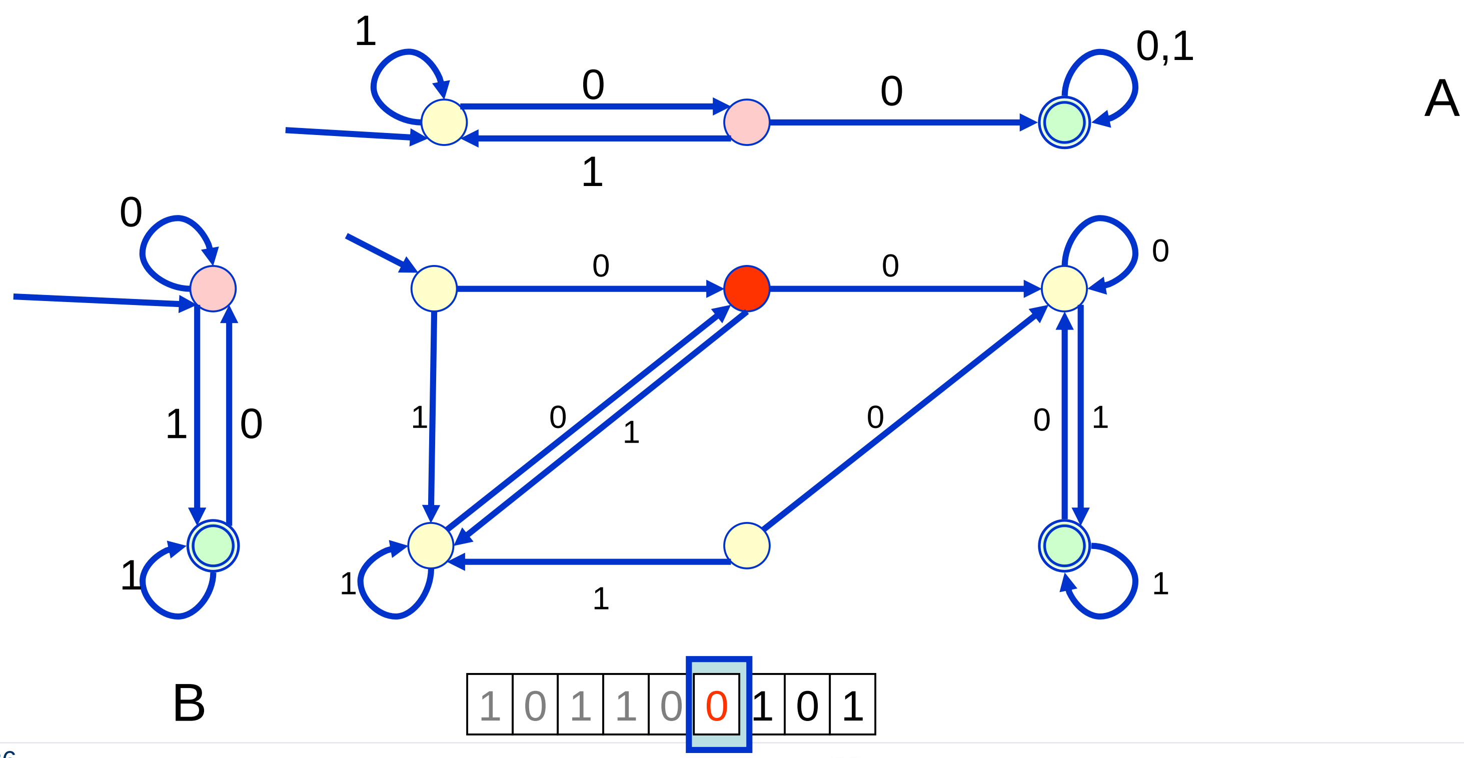


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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

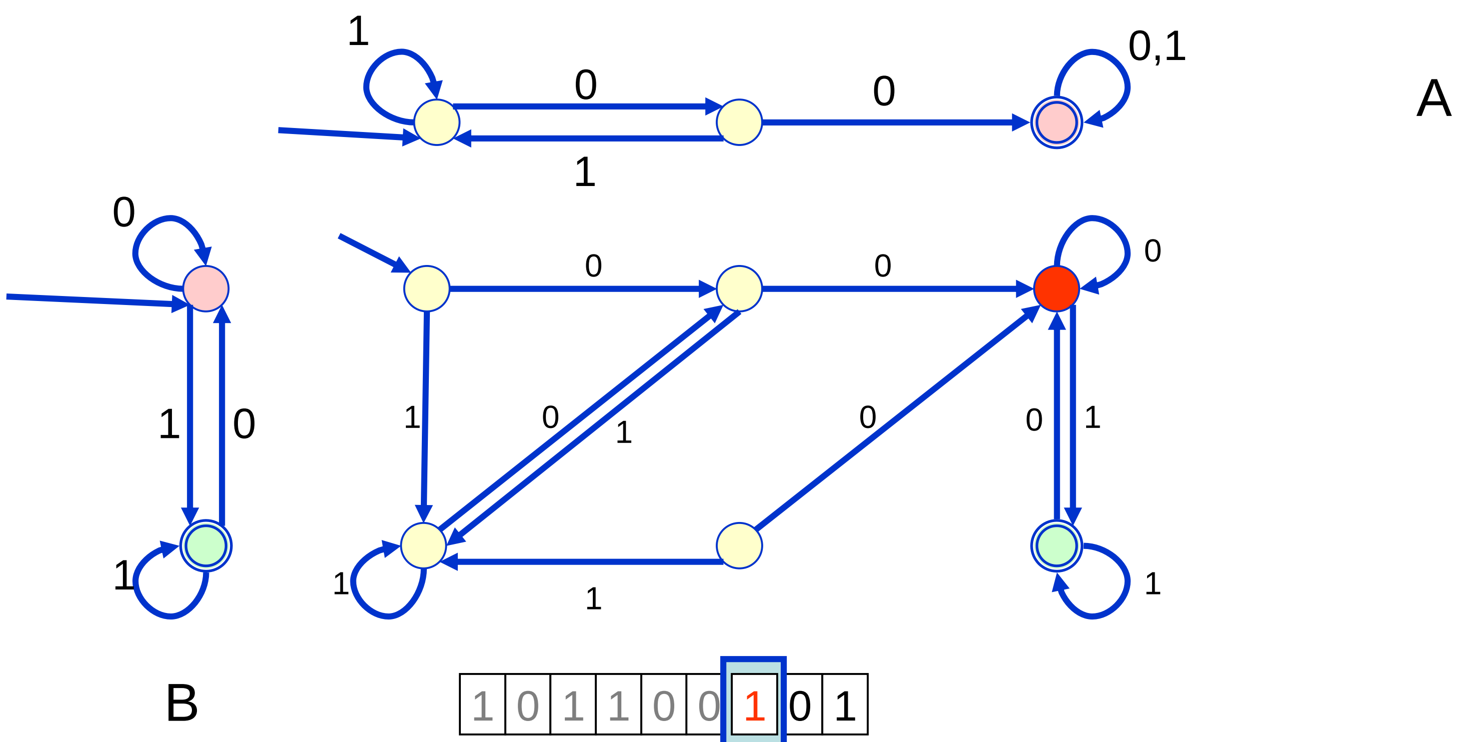


Produktautomat

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

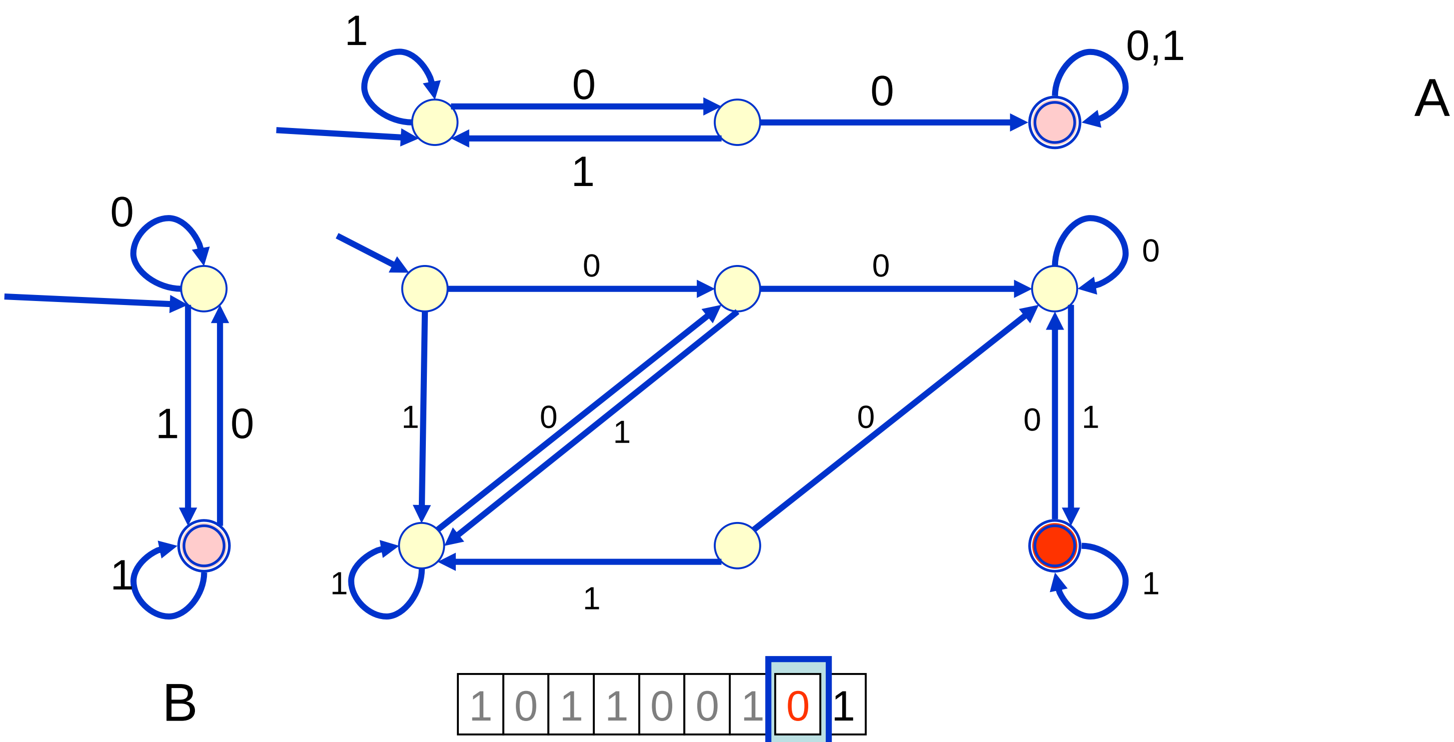


Produktautomat

$A = (P, \Sigma, \delta_A, p_0, F_A)$ 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

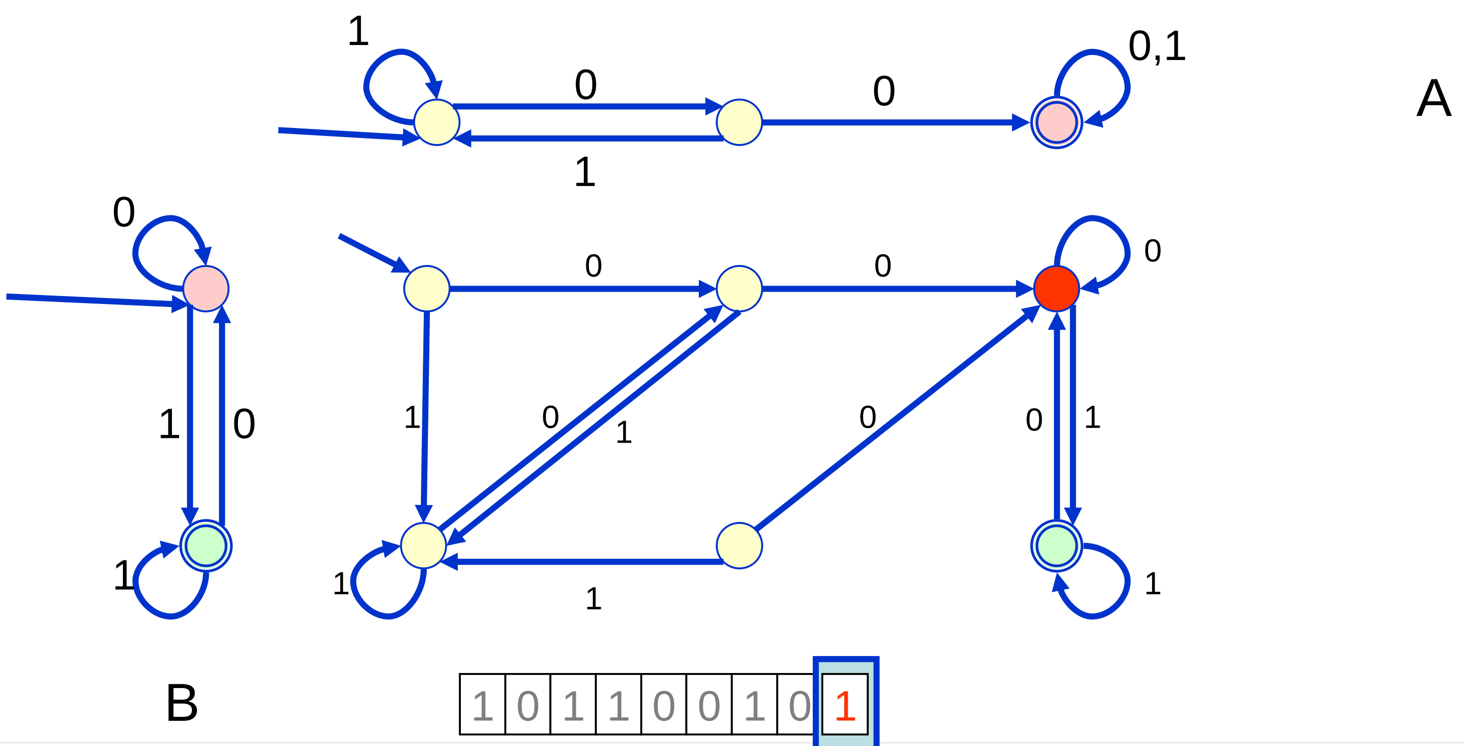


Produktautomat

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

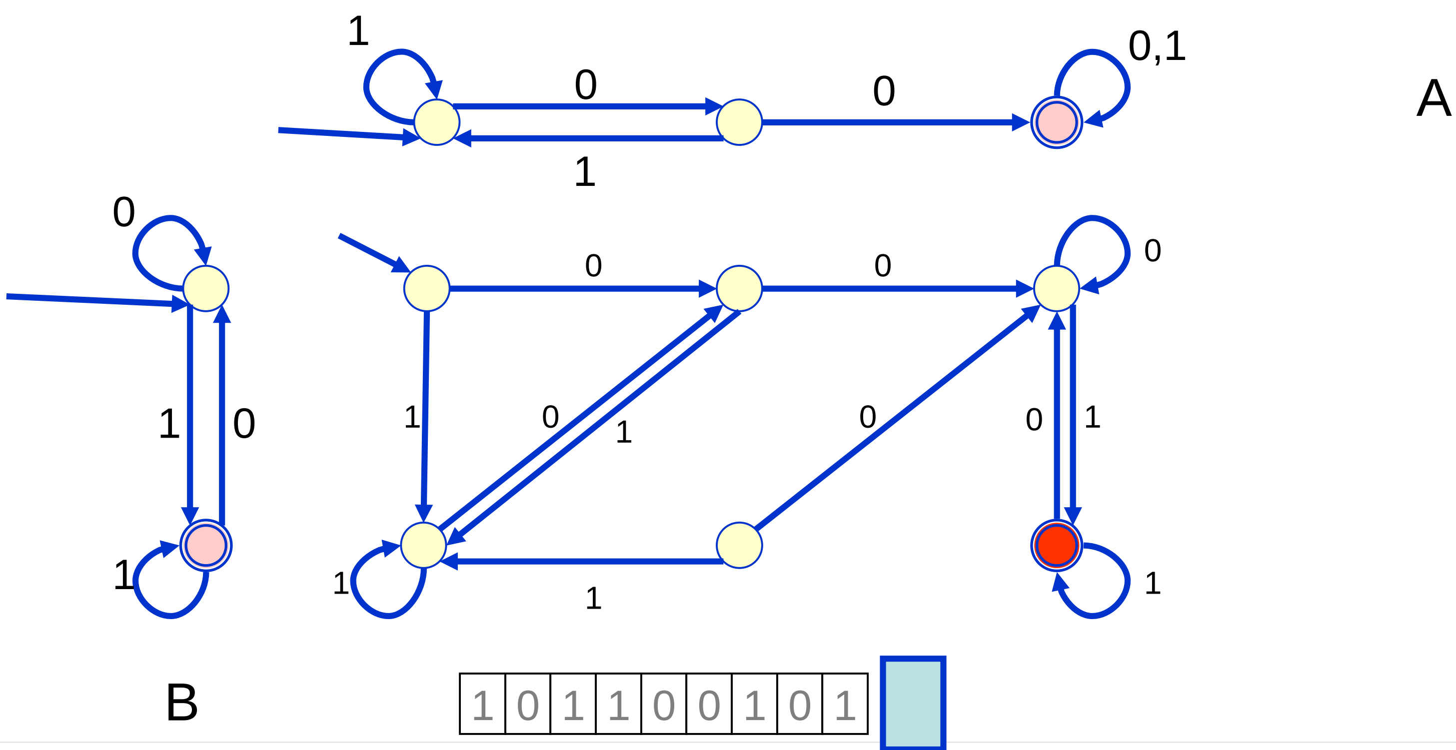


Produktautomat

$A = (P,\Sigma,\delta_A,p_0,F_A)$ 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



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)

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

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

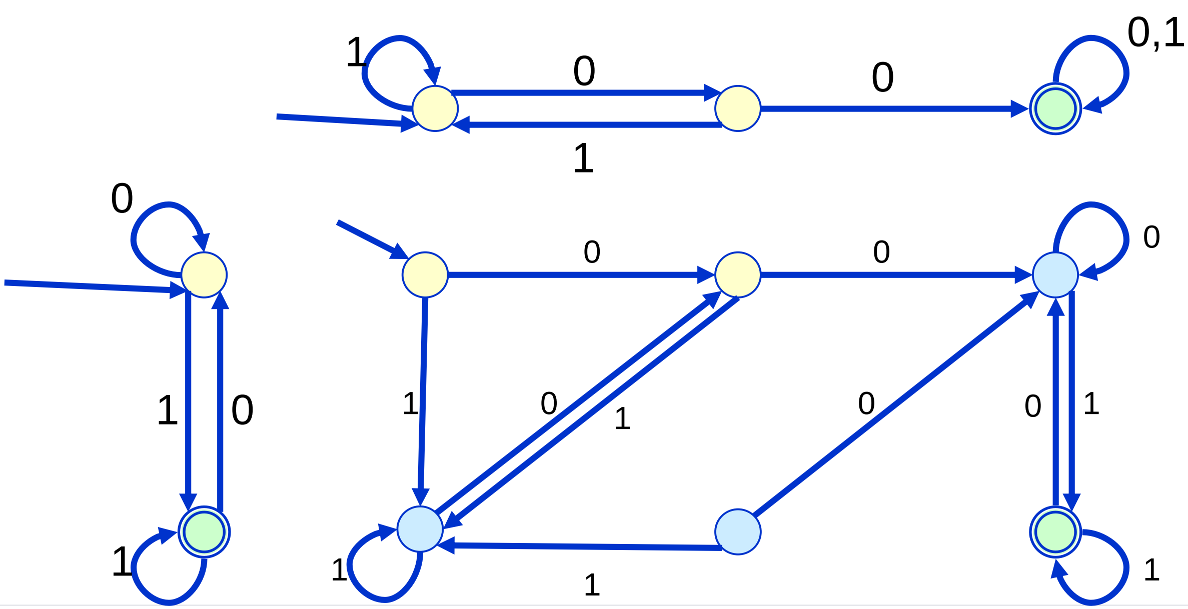
Parallelität und Produktautomat

- Parallele Komposition
  - A und B laufen gleichzeitig
  - synchron

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

Produktautomat

„Summenautomat“



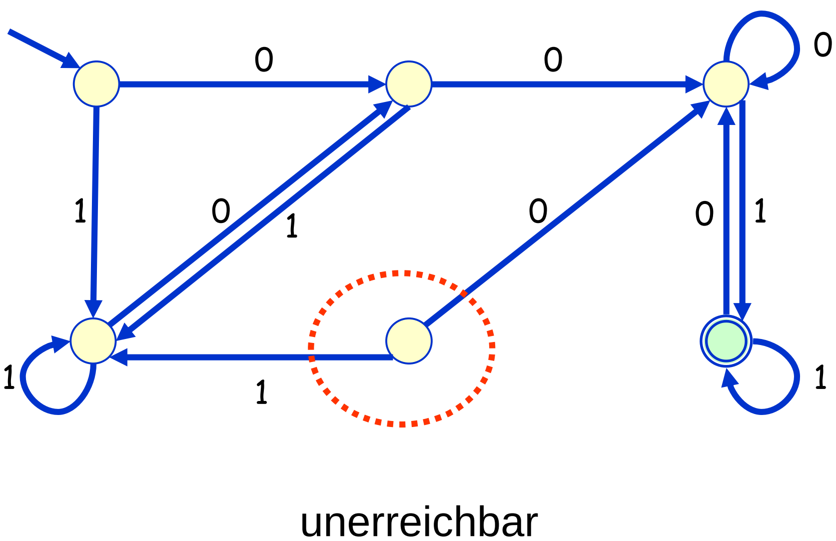
A

B

Produktautomat

Produktkonstruktion liefert nicht unbedingt den einfachsten Automat

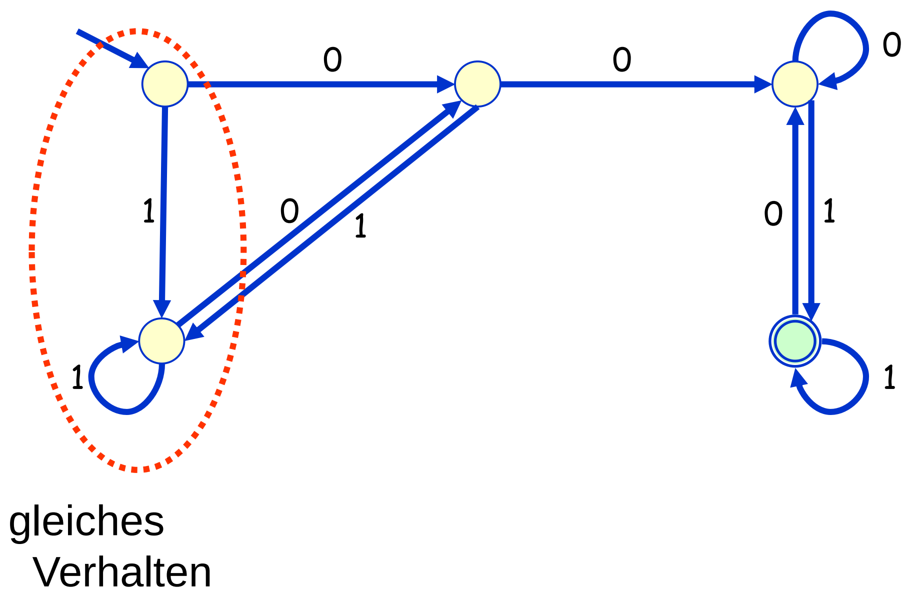
- Hier:
  - ein Zustand nicht erreichbar - entfernen



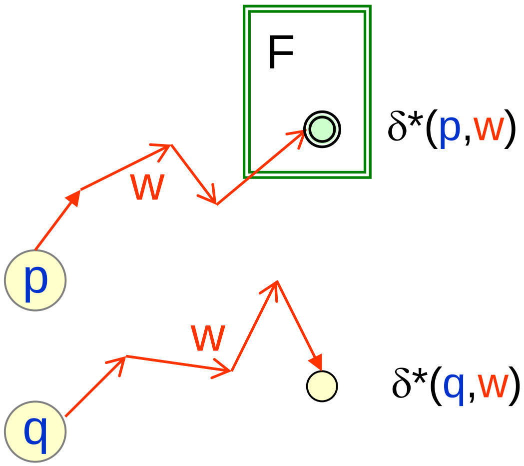
Produktautomat

Produktkonstruktion liefert nicht unbedingt den einfachsten Automaten

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



(Nicht) trennbare Zustände



Zustände p und q heißen trennbar,
wenn es ein Wort w gibt mit

$$\delta^*(p,w) \in F \text{ aber } \delta^*(q,w) \notin F,$$

oder

$$\delta^*(p,w) \notin F \text{ aber } \delta^*(q,w) \in F.$$

p und q heißen verhaltensgleich oder ununterscheidbar und wir schreiben

$$p \sim q$$

falls p und q **nicht** trennbar sind, also

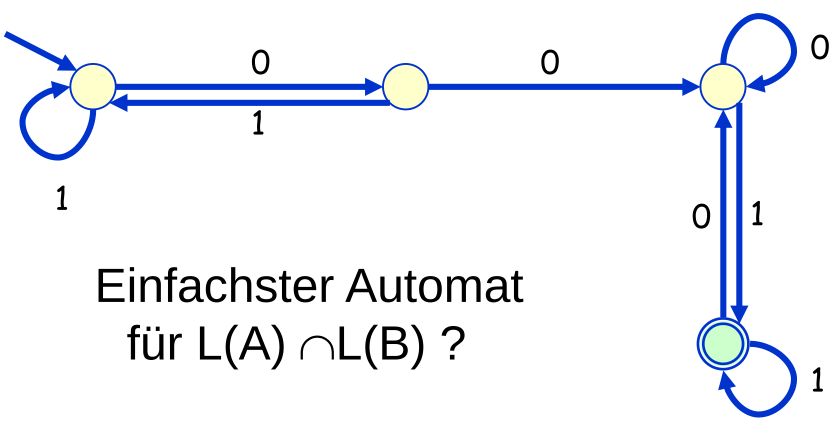
– $p \sim q \;\Leftrightarrow\; \forall\, w \in \Sigma^*: (\, \delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F \,)$

- $\sim$ ist eine Äquivalenzrelation

Produktautomat

Produktkonstruktion liefert nicht unbedingt den einfachsten Automat

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



Einfachster Automat
für L(A) $\cap$L(B) ?

Eigenschaften von ~

Def: $p \sim q \;\Leftrightarrow\; \forall w \in \Sigma^*: (\, \delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F \,)$

- Satz: Für alle Zustände p,q gilt
  - $p \sim q \;\Leftrightarrow$ 1. $(p \in F \Leftrightarrow q \in F)$ und
    2. $\forall a \in \Sigma:\; \delta(p,a) \sim \delta(q,a)$

- Beweis: „$\Rightarrow$“: Aus p~q folgt
  - 1. $p \in F \;\Leftrightarrow\; \delta^*(p,\varepsilon) \in F \;\Leftrightarrow\; \delta^*(q,\varepsilon) \in F \;\Leftrightarrow\; q \in F$
  - 2. Sei $\delta^*(\delta(p,a),w) \in F$ dann gilt $\delta^*(p,a.w) \in F$,
    somit $\delta^*(q,a.w) \in F$, also $\delta^*(\delta(q,a),w) \in F$.
    Folglich ist $\delta(p,a) \sim \delta(q,a)$.

  - „$\Leftarrow$“: Fallunterscheidung: $w = \varepsilon$ und w=a.u.
- Verhaltensgleichheit (~) ist eine Äquivalenzrelation
  - Beweis: Übung

Faktorautomat

- $A/\sim$ entsteht aus $A=(Q,\Sigma,\delta,q_0,F)$ indem man verhaltensgleiche Zustände identifiziert:
- $A/\sim \quad := \; (Q/\sim,\Sigma,\delta_\sim,[q_0]_\sim, F_\sim)$

mit

- $Q/\sim \quad := \{\, [q]_\sim \mid q\in Q \,\}$ mit $[q]_\sim := \{q' \in Q \mid q'\sim q\}$
- $\delta_\sim([q]_\sim,a) \quad := [\, \delta(q,a) \,]_\sim$
- $F_\sim \quad := \{[q]_\sim \mid q\in F \,\}$

- Wichtig: $F_\sim$ und $\delta_\sim$ sind wohldefiniert wegen

  $[q]_\sim = [q']_\sim \Rightarrow$
  1. $q\in F \Leftrightarrow q'\in F$
  2. $[\delta(q,a)]_\sim = [\delta(q',a)]_\sim$

Nach Konstruktion gilt:

In $A/\sim$ sind je zwei verschiedene „Zustände“ trennbar.

Beweis: Nach Definition: $p \sim q \;\Leftrightarrow\; \forall\, w\in\Sigma^*: (\, \delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F \,)$.
$[p]_\sim \neq [q]_\sim$ heißt $\neg(p \sim q)$, also $\exists w\in\Sigma^*: (\, \delta_\sim^*(p,w) \in F_\sim \Leftrightarrow \delta_\sim^*(q,w) \notin F_\sim \,)$.

Homomorphismen

- $A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$ seien Automaten.
- Eine Abbildung $\varphi : P \to Q$ heißt Homomorphismus, falls
  - i. $\varphi(p_0) = q_0$ // φ erhält … Anfangszustand
  - ii. $p \in F_A \Leftrightarrow \varphi(p) \in F_B$. // … akzeptierende Zustände
  - iii. $\varphi(\delta_A(p,e)) = \delta_B(\varphi(p),e)$ für alle $p \in P$ und alle $e \in \Sigma$ // …Transitionen

<u>Lemma</u>: Für jeden Homomorphismus φ gilt auch

$\varphi(\delta^*(p,w)) = \delta^*(\varphi(p),w)$ für alle $p \in P$ und alle $w \in \Sigma^*$.

<u>Beweis</u>: Induktion über w.

<u>Satz</u>: Ist $\varphi : A \to B$ ein Homomorphismus, so gilt $L(A) = L(B)$.

<u>Beweis</u>: $w \in L(A) \Leftrightarrow \delta^*(p_0,w) \in F_A$

$\Leftrightarrow \varphi(\delta^*(p_0,w)) \in F_B$ // wegen ii

$\Leftrightarrow \delta^*(\varphi(p_0),w) \in F_B$ // wegen dem Lemma

$\Leftrightarrow \delta^*(q_0,w) \in F_B$ // wegen i

$\Leftrightarrow w \in L(B)$ // Def L(B)

Satz vom Faktorautomaten

Satz: Die Abbildung $\pi_\sim : A \rightarrow A/\sim$ mit

$$\pi_\sim(p) := [p]_\sim$$

ist ein Homomorphismus.

Beweis:

i. $\pi_\sim(p_0) = [p_0]_\sim$ ist nach Definition initial in $A/\sim$

ii. $p \in F_A \Leftrightarrow \pi_\sim(p) \in F_\sim$ wegen der Definition: $F_\sim := \{[p]_\sim \mid p \in F\}$

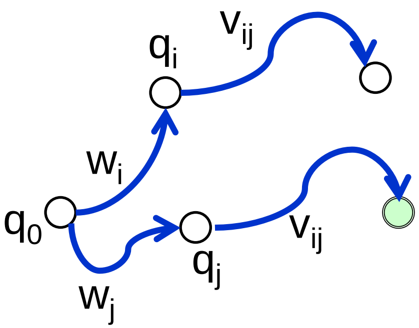
iii. $\pi_\sim(\delta_A(p,e)) = [\delta_A(p,e)]_\sim = \delta_\sim([p]_\sim,e) = \delta_\sim(\pi_\sim(p),e)$

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

Minimalautomat

Ein Automat A heißt minimal, falls

- jeder Zustand erreichbar ist
- je zwei verschiedene Zustände trennbar sind



Zu jedem Automaten A gibt es einen minimalen Automaten, der die gleiche Sprache erkennt.

Beweis: Gegeben A, entferne alle nicht erreichbaren Zustände. Der entstandene Automat A' erkennt die gleiche Sprache, also L(A) = L(A'). Der Faktorautomat A'/~ ist minimal und es gilt L(A'/~) = L(A') = L(A).

Eindeutigkeit des Minimalautomaten (1)

Satz: Seien $A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$

a. minimale Automaten mit
b. $L(A)=L(B)$.

Dann gibt es einen bijektiven Homomorphismus $\varphi : A \to B$

Lemma: Für beliebige Worte $u,v\in\Sigma^*$ gilt:

$$\delta_A^*(p_0,u) = \delta_A^*(p_0,v) \Leftrightarrow \delta_B^*(q_0,u) = \delta_B^*(q_0,v).$$

Beweis des Lemmas: Wegen der Trennbarkeit von A und von B folgt

$\delta_A^*(p_0,u) = \delta_A^*(p_0,v) \Leftrightarrow \delta_A^*(p_0,u) \sim \delta_A^*(p_0,v)$ // A trennbar

$\Leftrightarrow \forall w\in\Sigma^*: \delta_A^*(\delta_A^*(p_0,u),w)\in F_A$ gdw. $\delta_A^*(\delta_A^*(p_0,v),w) \in F_A$ // Def ~

$\Leftrightarrow \forall w\in\Sigma^*: uw\in L(A)$ gdw $vw\in L(A)$ // $\delta_A^*(\delta_A^*(p,u),w) = \delta_A^*(p,uw)$

$\Leftrightarrow \forall w\in\Sigma^*: uw\in L(B)$ gdw. $vw\in L(B)$ // $L(A) = L(B)$

$\Leftrightarrow \delta_B^*(q_0,u) = \delta_B^*(q_0,v)$ // Def $L(B)$

Eindeutigkeit des Minimalautomaten (2)

Satz: Seien $A = (P,\Sigma,\delta_A,p_0,F_A)$ und $B = (Q,\Sigma,\delta_B,q_0,F_B)$

a. minimale Automaten mit
b. L(A)=L(B).

Dann gibt es einen bijektiven Homomorphismus $\varphi : A \to B$

Lemma: Für beliebige Worte $u,v \in \Sigma^*$ gilt:

$$\delta_A^*(p_0,u) = \delta_A^*(p_0,v) \Leftrightarrow \delta_B^*(q_0,u) = \delta_B^*(q_0,v).$$

Beweis des Satzes: Jedes $p \in P$ ist erreichbar, d.h. $\forall p \in P: \exists u \in \Sigma^*: p = \delta_A^*(p_0,u)$. Setze

$$\varphi(p) := \delta_B^*(q_0,u).$$

Wegen des Lemmas ist φ wohldefiniert und injektiv. Weil jeder Zustand in Q erreichbar ist, ist φ auch surjektiv, also bijektiv.

φ ist ein Homomorphismus, weil für jedes $p = \delta_A^*(p_0,u)$ gilt:

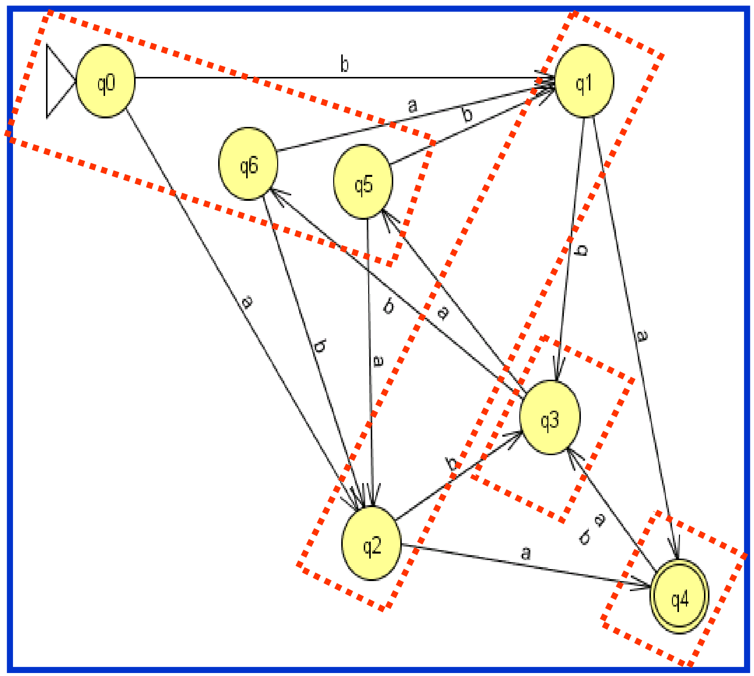
(i) $\varphi(p_0) = \varphi(\delta_A^*(p_0, \varepsilon)) = \delta_B^*(q_0, \varepsilon) = q_0$.

(ii) $\varphi(\delta_A(p,e)) = \varphi(\delta_A(\delta_A^*(p_0,u),e)) = \varphi(\delta_A^*(p_0,u.e)) = \delta_B^*(q_0,u.e) = \delta_B(\delta_B^*(q_0,u),e) = \delta_B(\varphi(p),e)$.

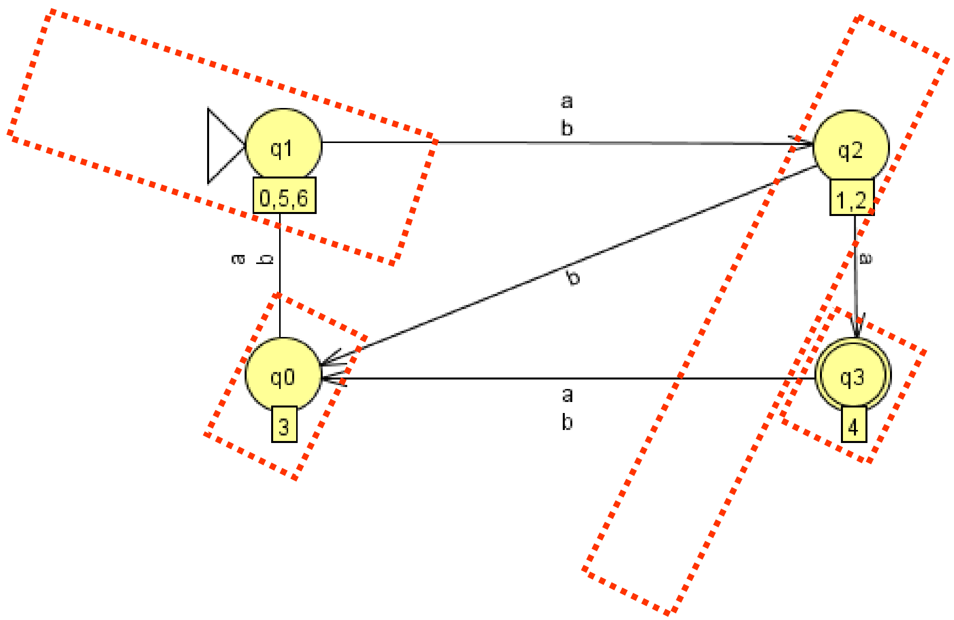
(iii) $p \in F_A \Leftrightarrow \exists w \in L(A): p = \delta_A^*(p_0,w) \Leftrightarrow \exists w \in L(B): \varphi(p) = \delta_B^*(q_0,w) \Leftrightarrow \varphi(p) \in F_B$

Minimierung

- Wie kann man A/~ effizient konstruieren ?



A

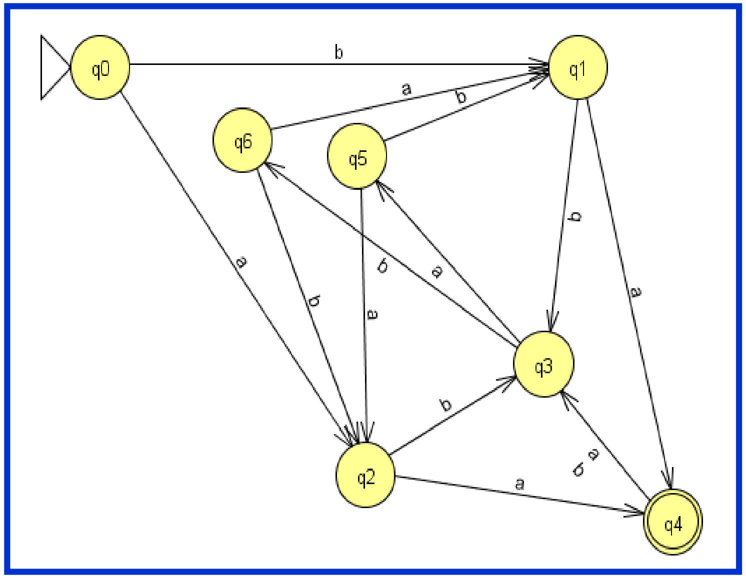


A/~

Minimierung

- Wie kann man A/~ effizient konstruieren ?

Maxime: trenne Zustände von A
nur wenn sie trennbar sind



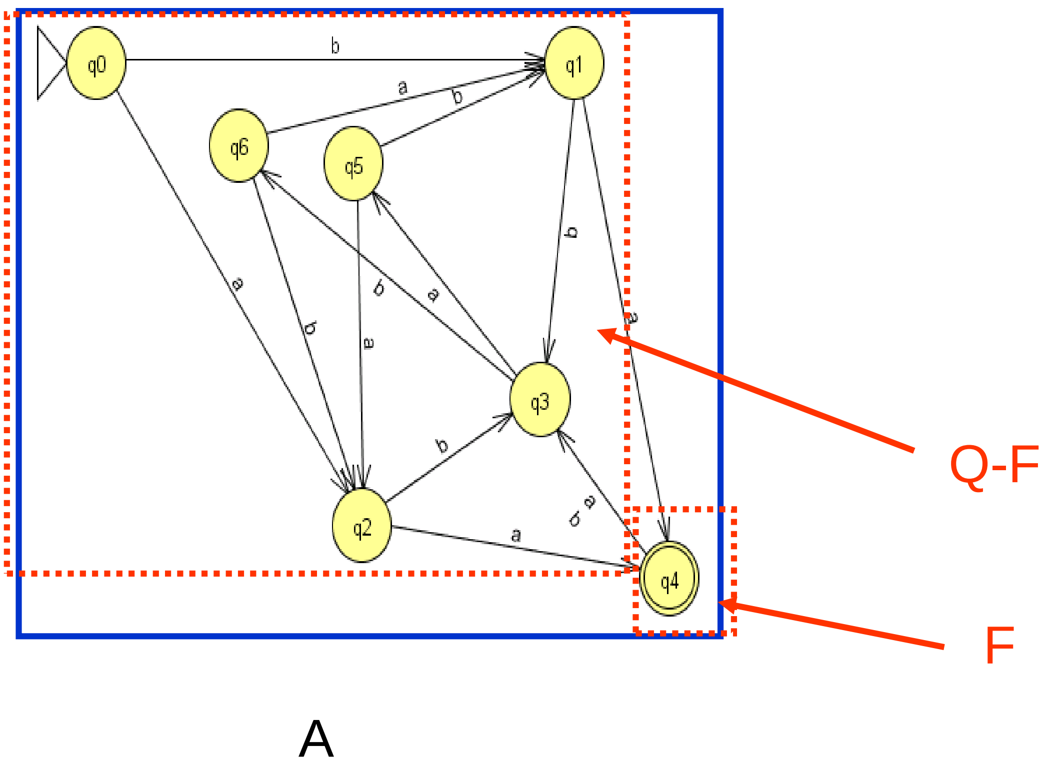
A

Minimierung

- Wie kann man A/~ effizient konstruieren?

<u>Maxime</u>: trenne Zustände von A nur wenn notwendig

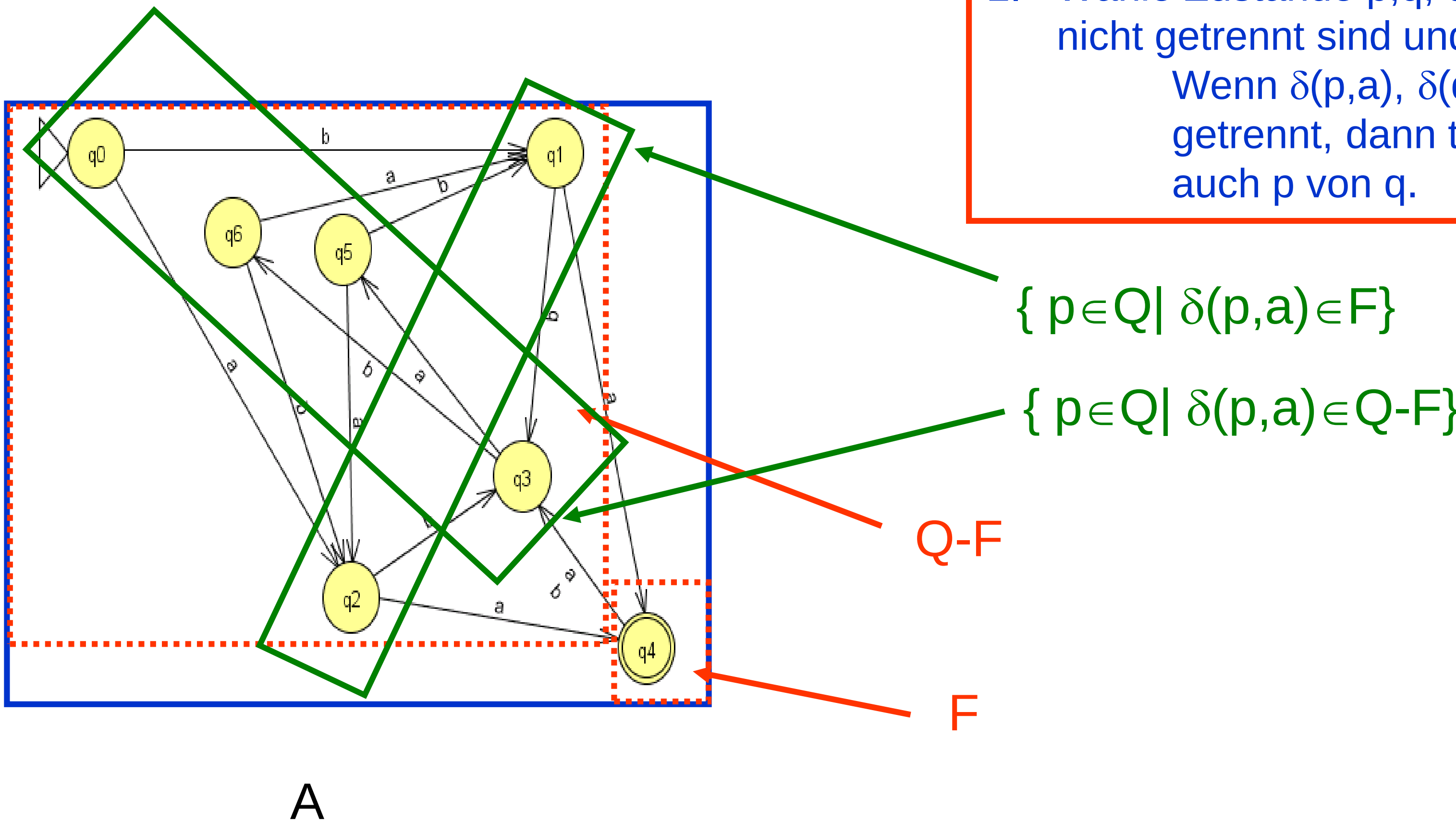
1. Trenne Endzustände von Nicht-Endzuständen



Philipps Universität Marburg

Minimierung

- Wie kann man A/~ effizient konstruieren?



<u>Maxime</u>: trenne Zustände von A nur wenn notwendig

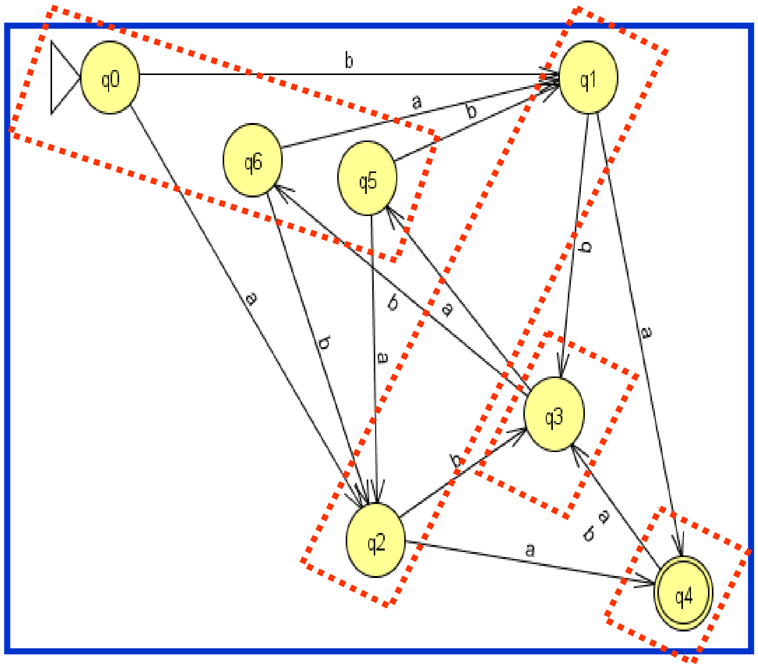
1. Trenne Endzustände von Nicht-Endzuständen

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.

Philipps Universität Marburg

Minimierung

- Wie kann man A/~ effizient konstruieren?



A

Maxime: trenne Zustände von A nur wenn notwendig

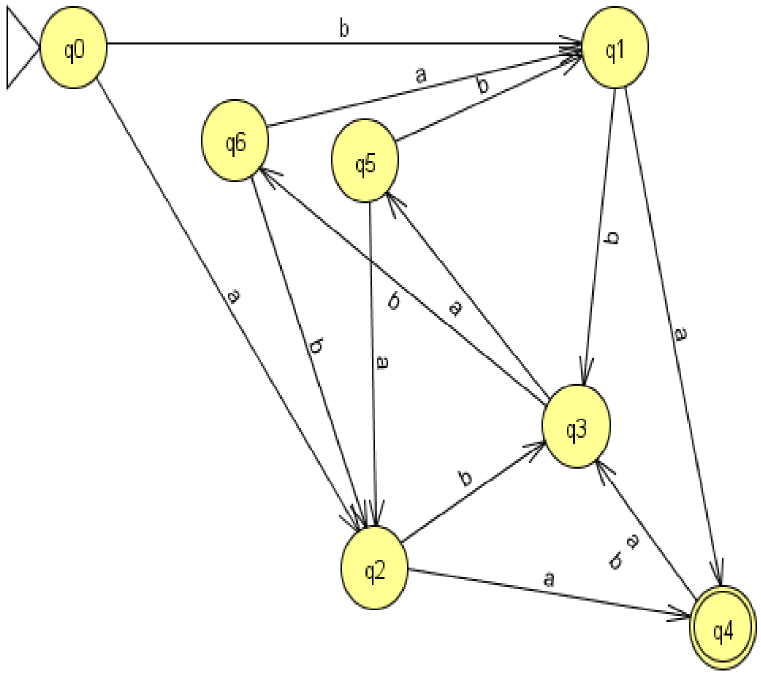
1. Trenne Endzustände von Nicht-Endzuständen

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

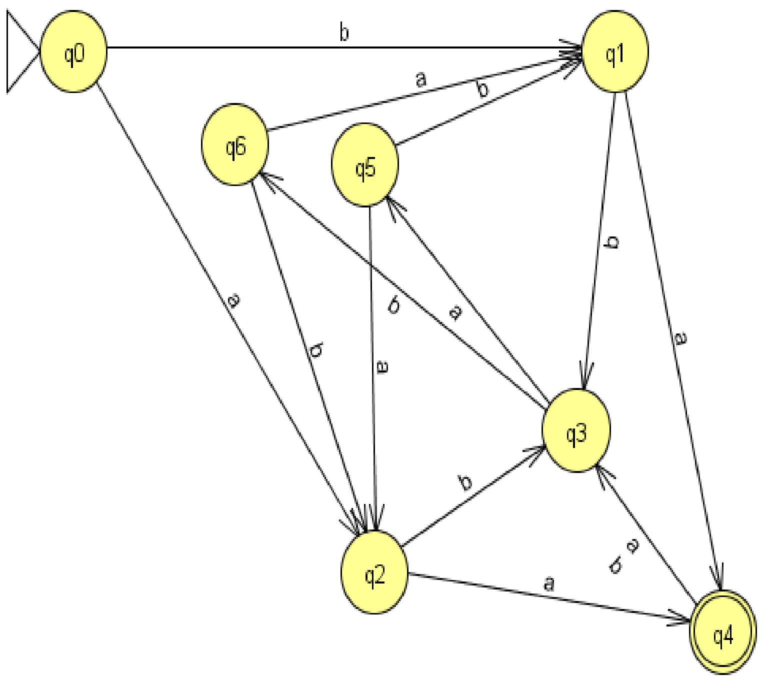
Darstellung in JFLAP

Partitionsbaum

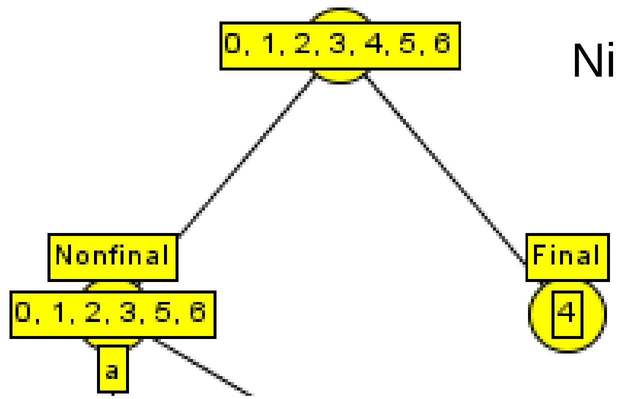
Nichts getrennt



Darstellung in JFLAP



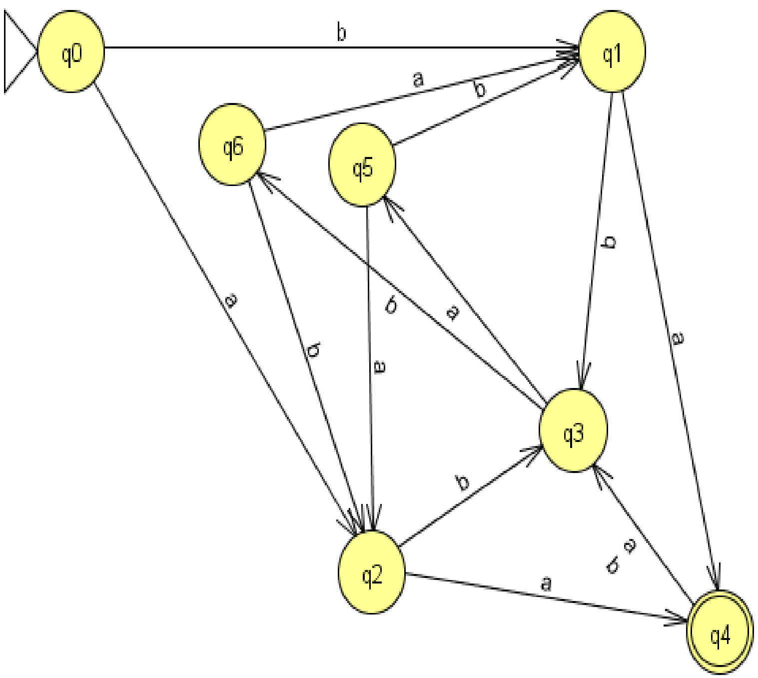
Partitionsbaum



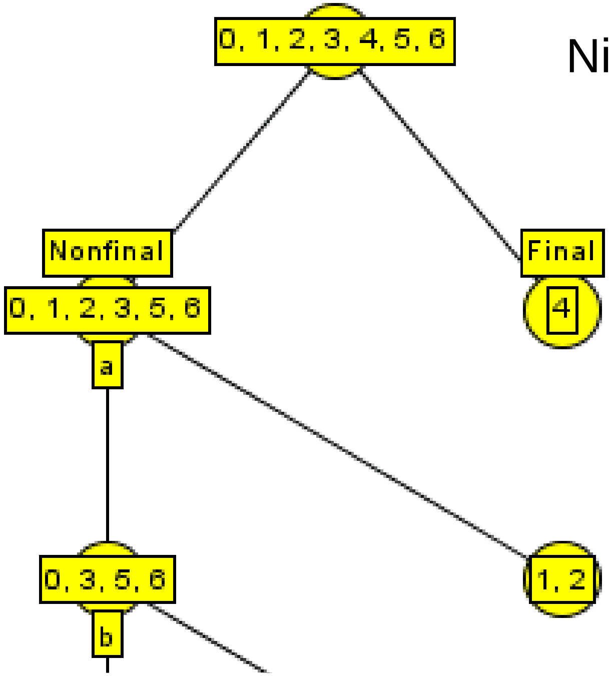
Nichts getrennt

Q-F von F

Darstellung in JFLAP



Partitionsbaum

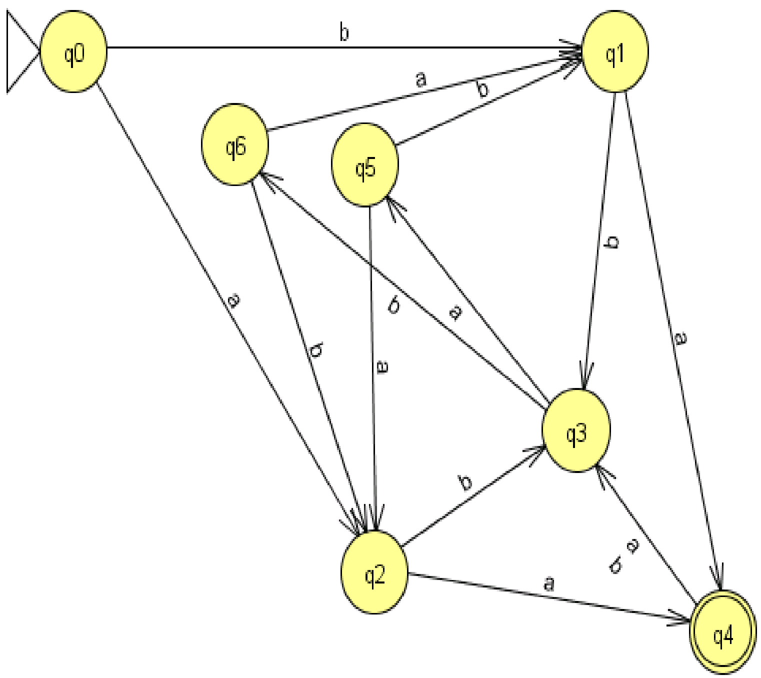


Nichts getrennt

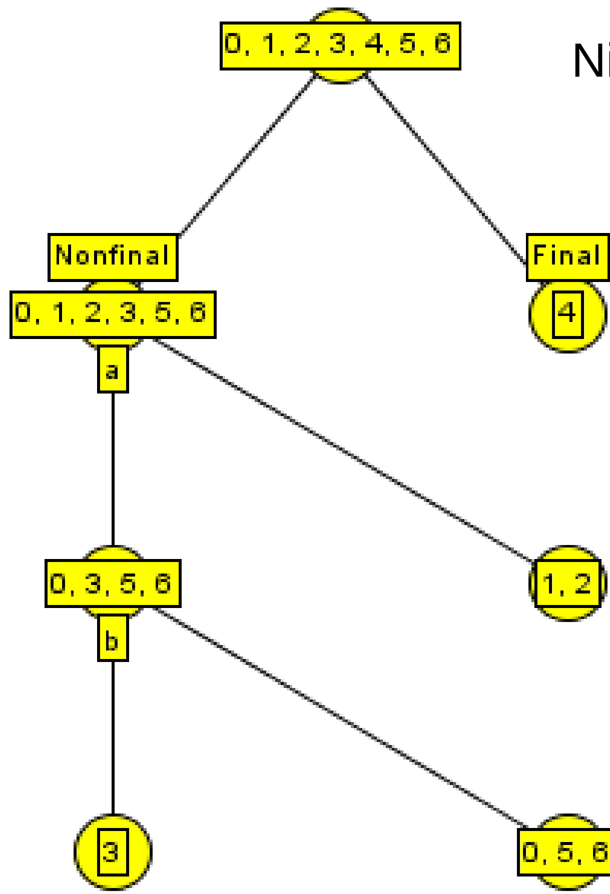
Q-F von F

Trennung nach a

Darstellung in JFLAP



Partitionsbaum



Nichts getrennt

Q-F von F

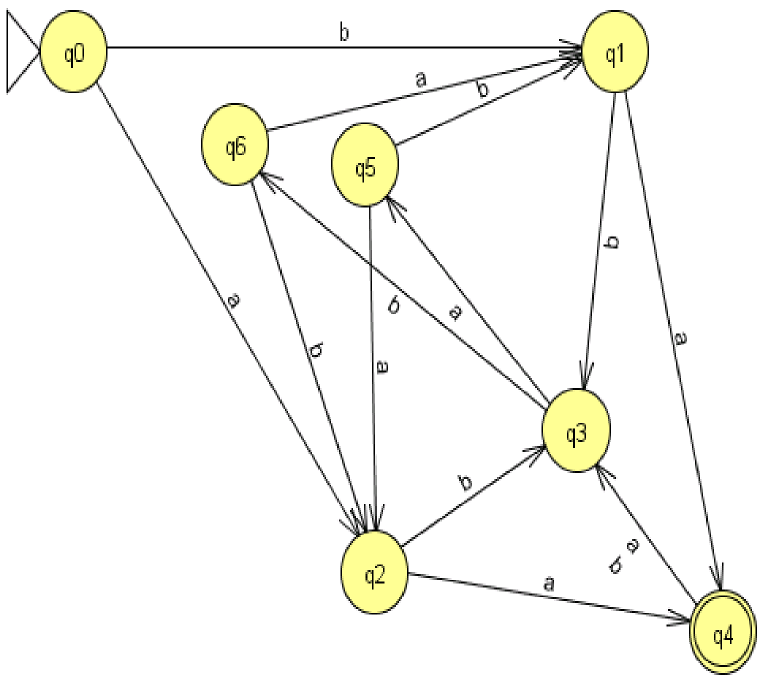
Trennung nach a

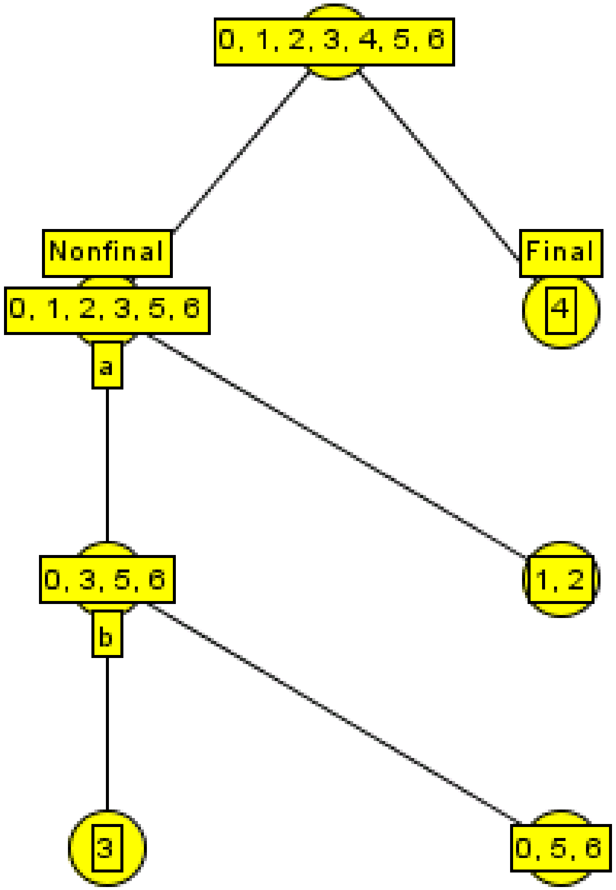
Trennung nach b

Schon fertig (Kleines Beispiel)

Fertiger Automat

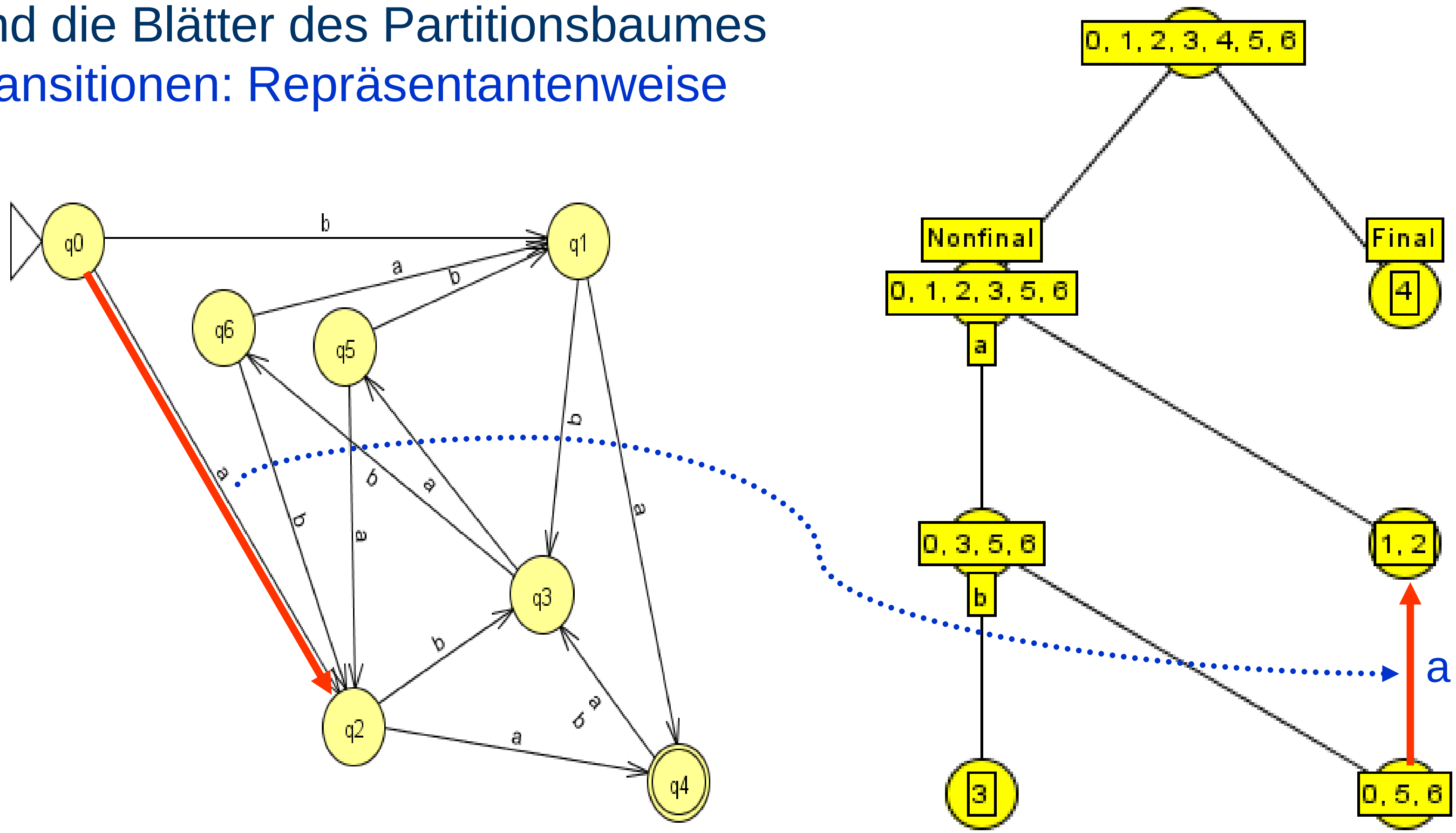
- Die Zustände des Minimalautomaten sind die Blätter des Partitionsbaumes
- Transitionen: Repräsentantenweise





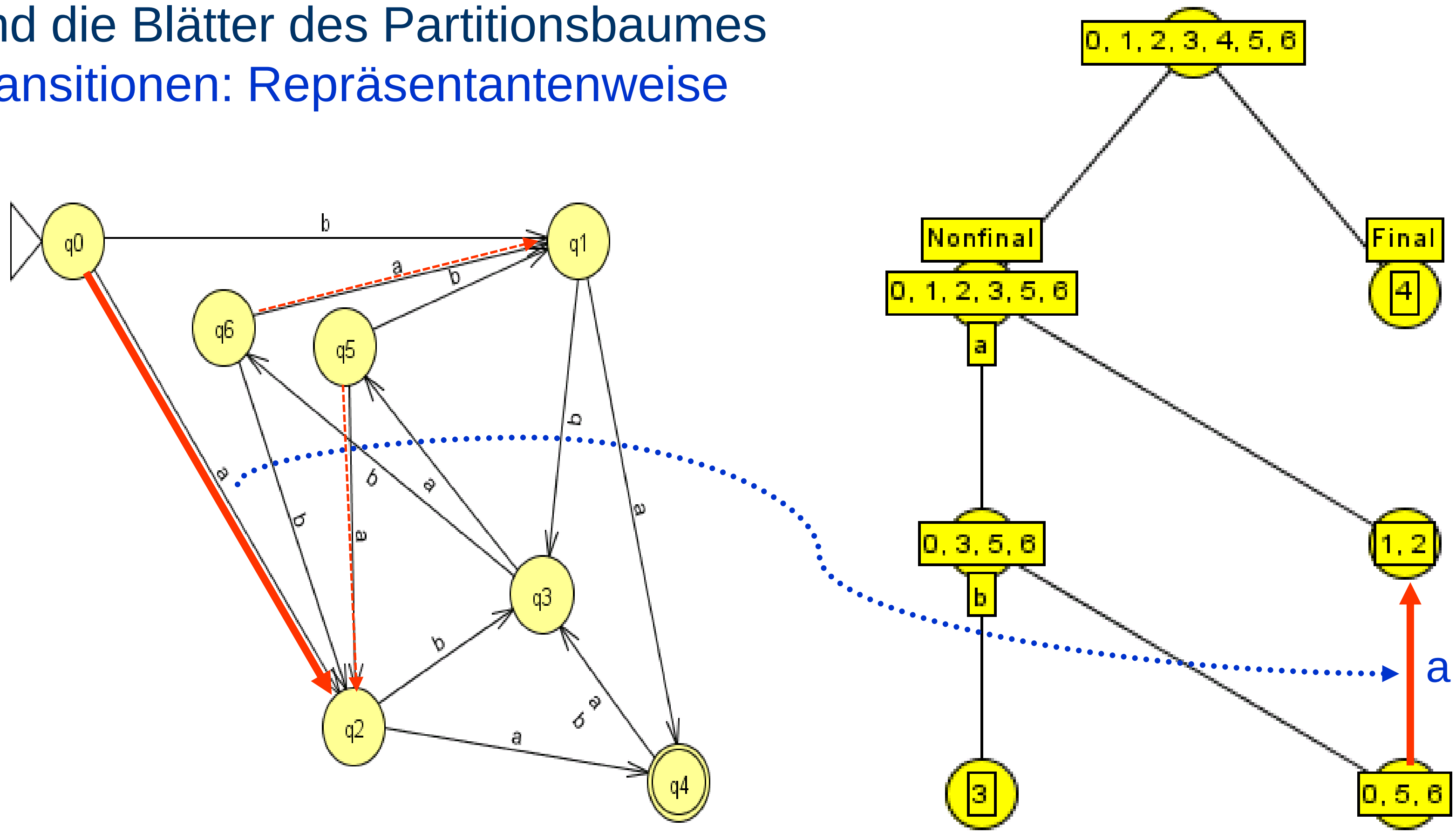
Fertiger Automat

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



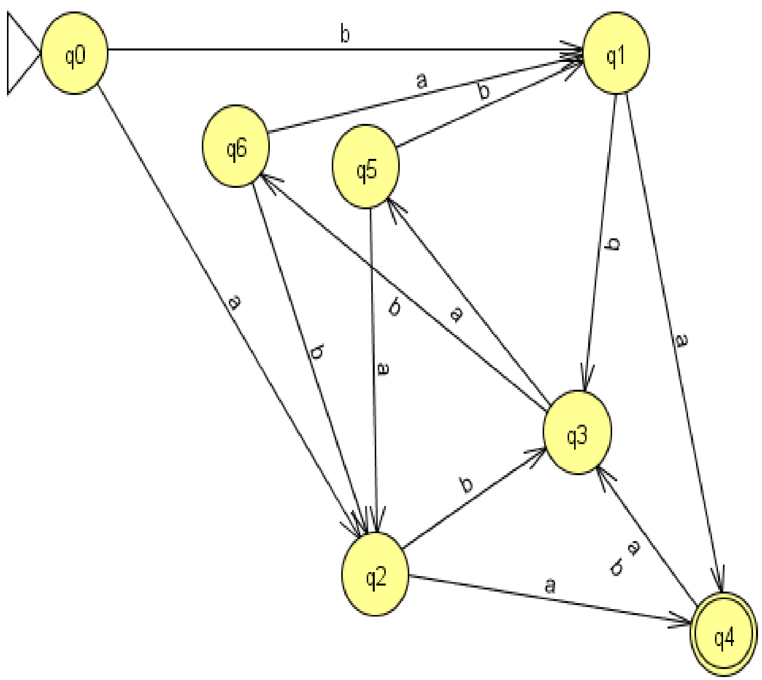
Fertiger Automat

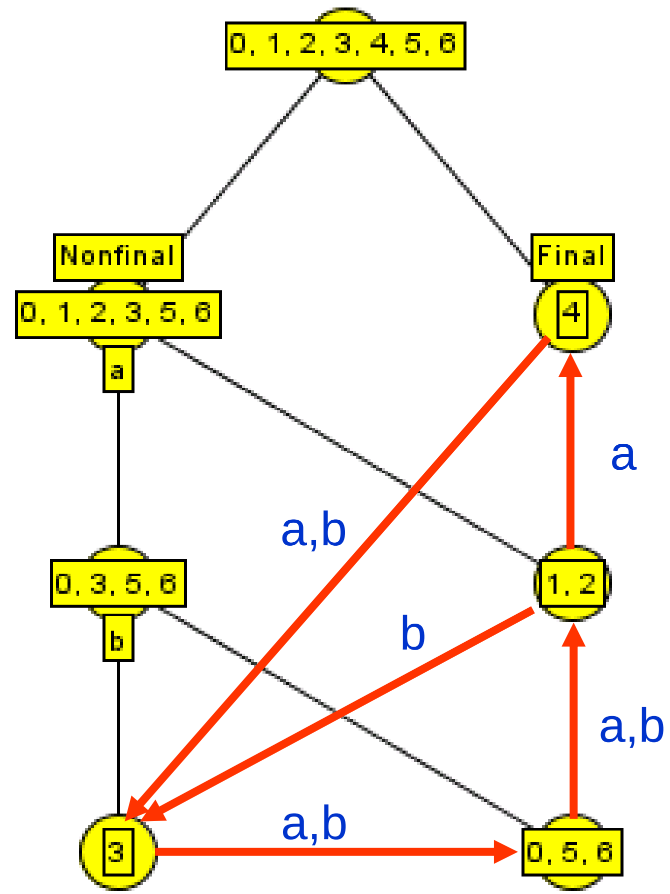
- Die Zustände des Minimalautomaten sind die Blätter des Partitionsbaumes
- Transitionen: Repräsentantenweise



Fertiger Automat

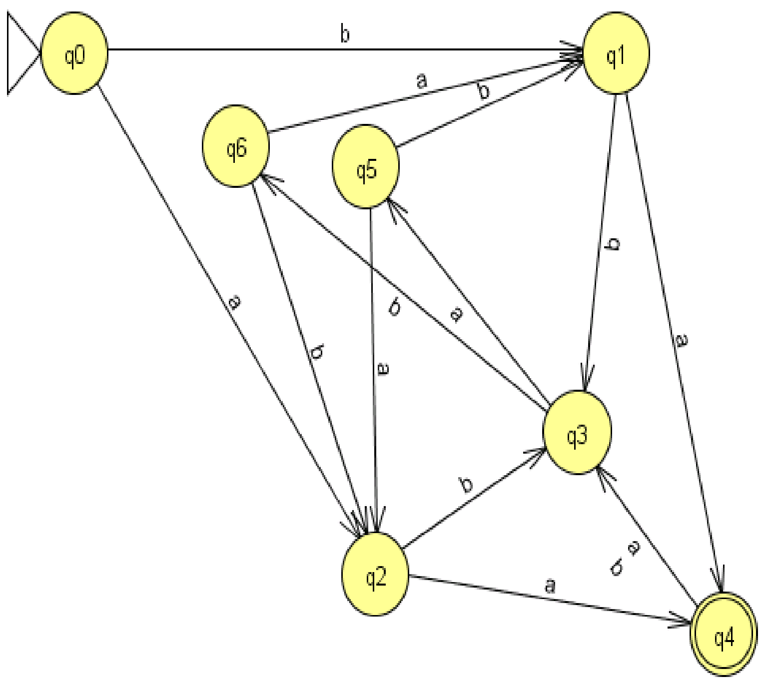
- Die Zustände des Minimalautomaten sind die Blätter des Partitionsbaumes
- Transitionen: Repräsentantenweise

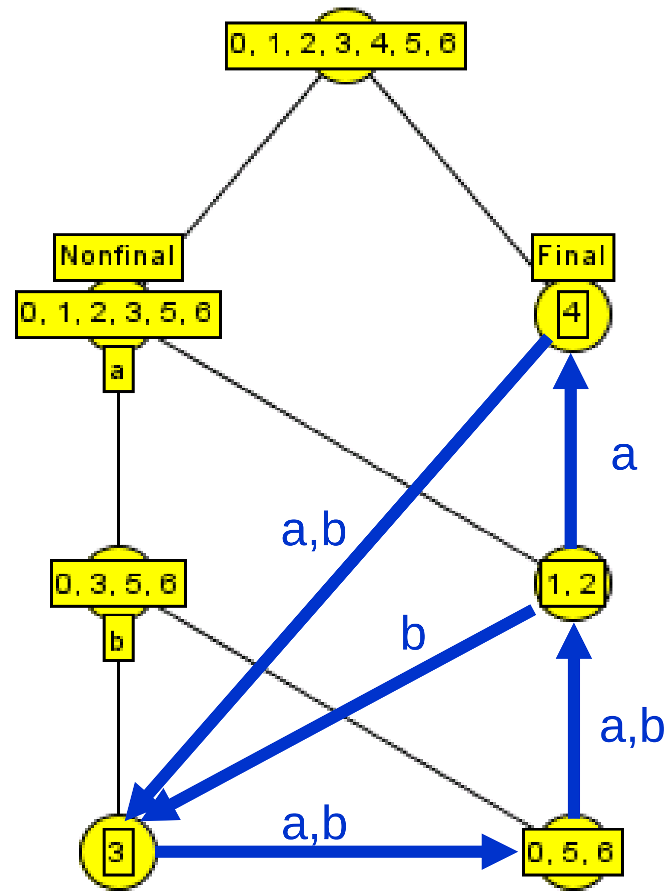




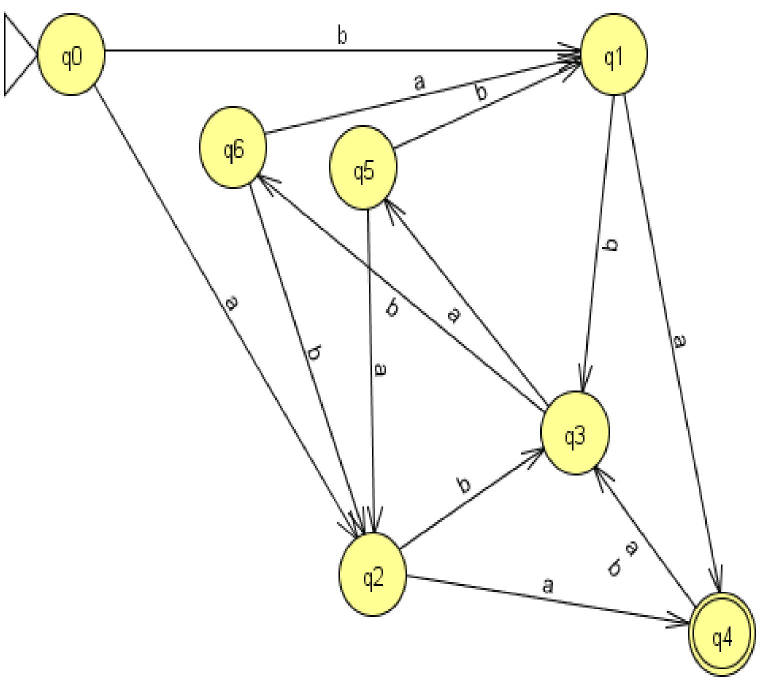
Fertiger Automat

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





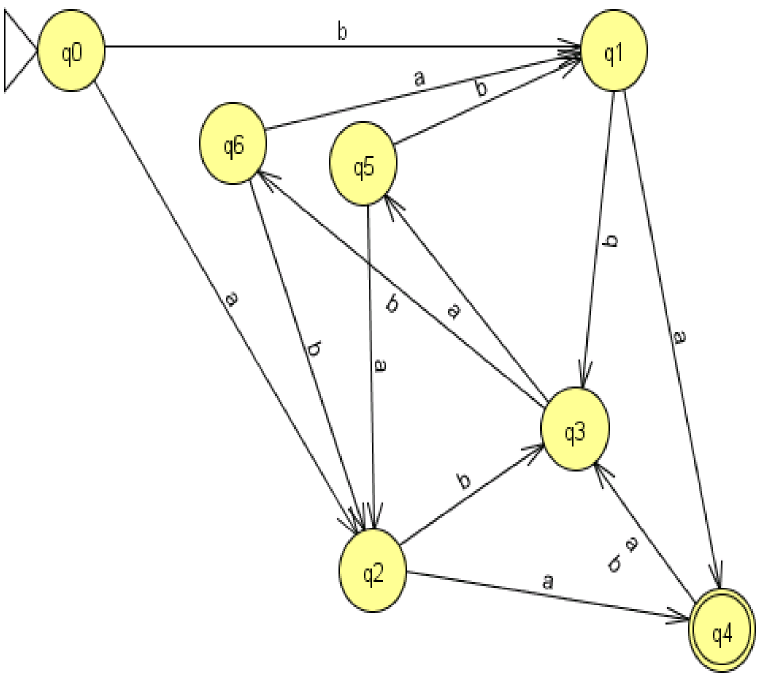
Andere Möglichkeit: Tabelle



- 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							
q_1							
q_2							
q_3							
q_4							
q_5							
q_6							

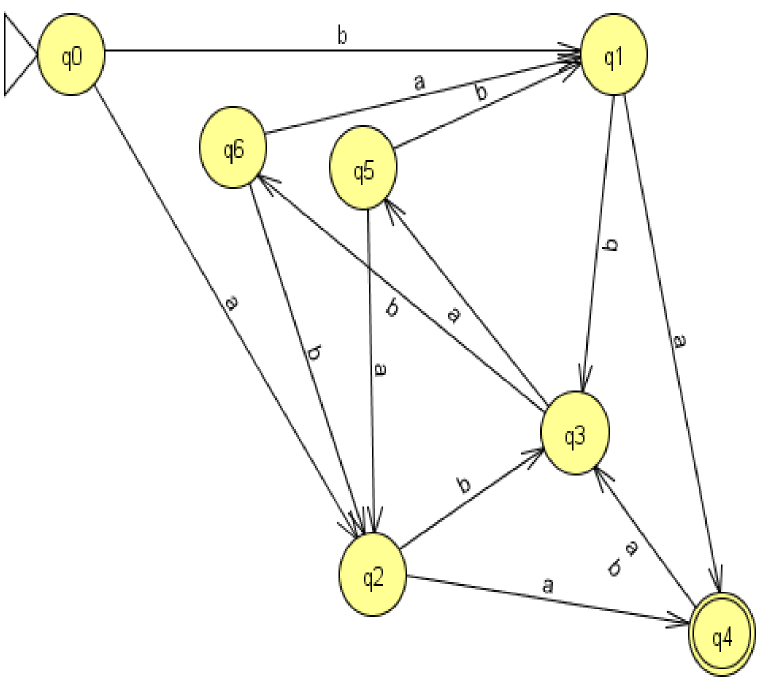
Andere Möglichkeit: Tabelle



- 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		
q_1					X		
q_2					X		
q_3					X		
q_4						X	X
q_5							
q_6							

Andere Möglichkeit: Tabelle



- Tabelliere Trennbarkeitsrelation
  - Trenne F und Q-F
  - Für alle e $\in\Sigma$,
    für alle p < q
    Falls δ(p,a), δ(q,a) schon getrennt,
    dann trenne p und q

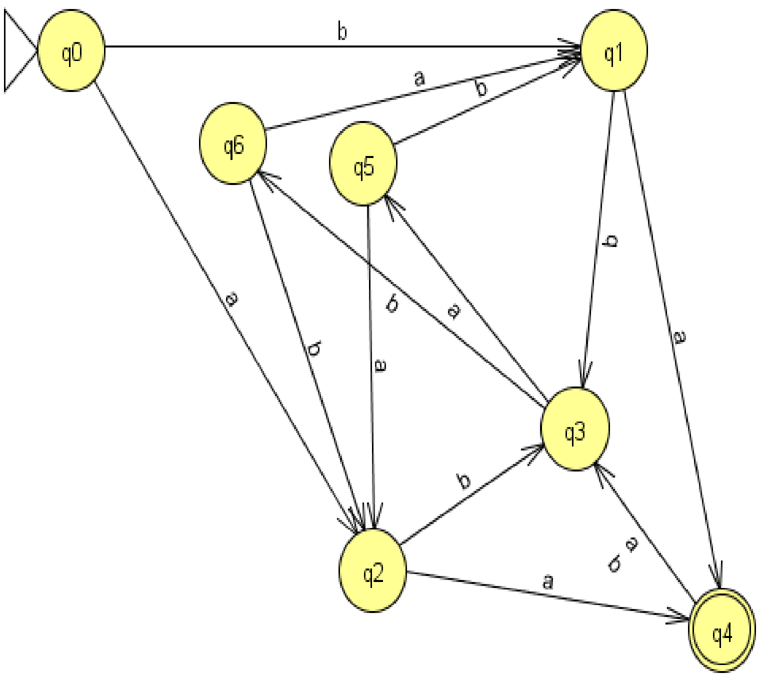
    $\boxed{e := a}$
  - Wiederhole bis eine Runde lang nichts getrennt wurde

q

p

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

Andere Möglichkeit: Tabelle

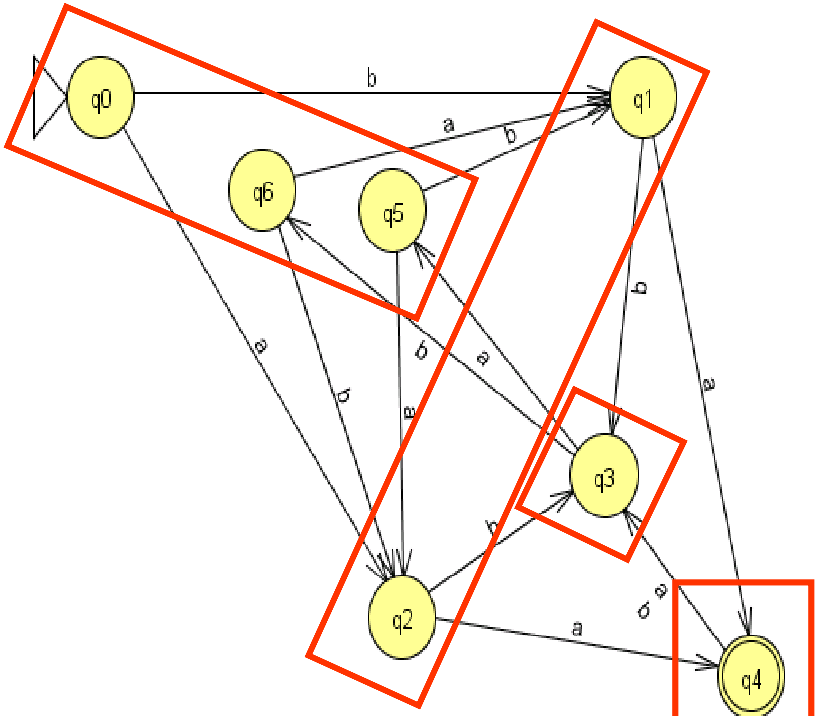


- Tabelliere Trennbarkeitsrelation
  - Trenne F und Q-F
  - Für alle e $\in\Sigma$,
    für alle p < q
    Falls $\delta(p,a)$, $\delta(q,a)$ schon getrennt,
    dann trenne p und q

    $\boxed{e := b}$
  - 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							

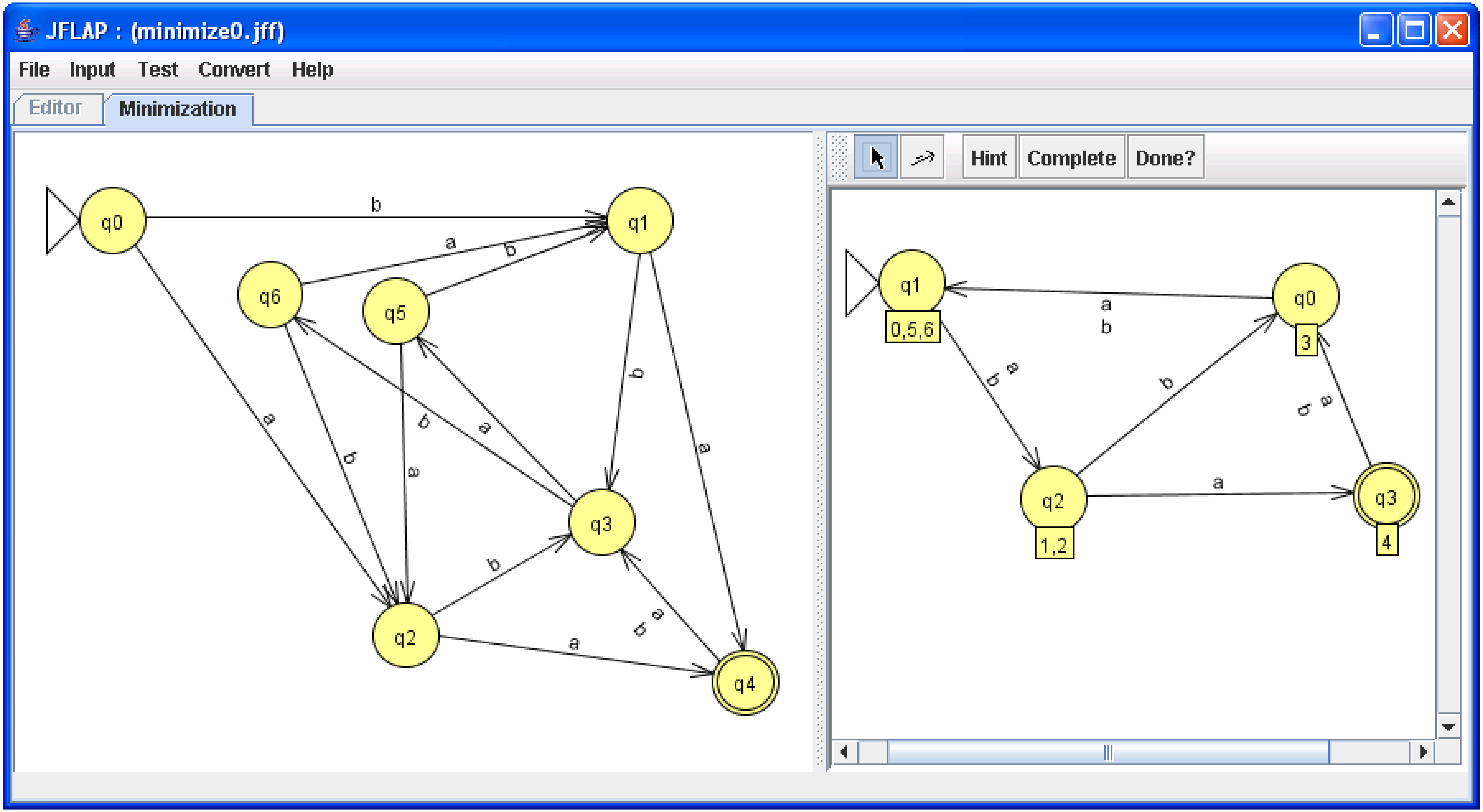
Andere Möglichkeit: Tabelle



- Tabelliere Trennbarkeitsrelation
  - Trenne F und Q-F
  - Für alle e $\in\Sigma$,
    für alle p < q
    Falls δ(p,a), δ(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


JFLAP : (minimize0.jff)
File Input Test Convert Help
Editor
Minimization
Hint
Complete
Done?
q0
q1
q2
q3
q4
q5
q6
a
b
0,5,6
3
1,2
4

Inhalt

1. Deterministische Akzeptoren
   - Grundbegriffe
   - Sprache eines Automaten
   - Implementierung
   - Komplement, Produkte
   - Faktorautomat
   - Minimalautomat

2. Grenzen von Automaten

   - Trennbarkeit
   - Nerode-Lemma
   - Pumping Lemma
   - Automaten mit Ausgabe

Die Grenzen von Automaten

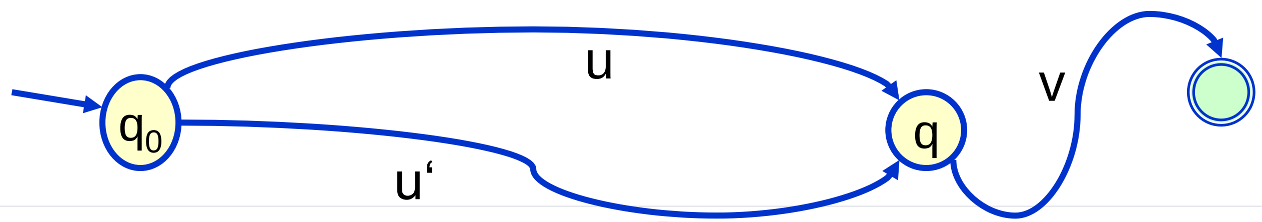
- Nicht jede Sprache $L \subseteq \Sigma^*$ lässt sich durch einen endlichen Automaten beschreiben.

  Beispiel: $L = \{ a^n b^n \mid n \geq 0 \}$

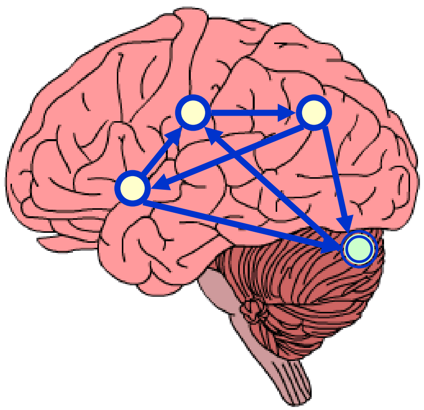
- Wir suchen zunächst intuitive Anhalltspunkte und gelangen dann zu handhabbaren exakten Kriterien

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 sich der 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 $k \in \mathbb{N}$ in einem anderen Zustand sein
- angenommen :
  - $\exists\, i \neq k: \delta^*(q_0, a^i) = q = \delta^*(q_0, a^k)$
    - einerseits : $\delta^*(q, b^i) = \delta^*(\delta^*(q_0, a^i), b^i) = \delta^*(q_0, a^i b^i) \in F$ sein
    - andererseits : $\delta^*(q, b^i) = \delta^*(\delta^*(q_0, a^k), b^i) = \delta^*(q_0, a^k b^i) \notin F$ 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:
  - $L = \{ a^i b^k \mid i \neq k \}$
    - 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
  - $L = \{ u^R u \mid u \in \Sigma^* \} \cup \{ u^R a u \mid a \in \Sigma, u \in \Sigma^* \}$ Palindrome
  - L = Menge aller wohlgeformten Klammerausdrücke
    - Dyck-Sprache
  - $L = \{ a^{n*n} \mid n \geq 0 \}$
    - klein bisschen schwieriger zu argumentieren
- Intuition liefert aber nur Anfangsverdacht
- Wir lernen zwei mathematische Beweismethoden kennen:
  - Nerode Lemma
  - Pumping Lemma

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: $L = \{ 0, 010, 0110, 01110, 011110, \dots \}$

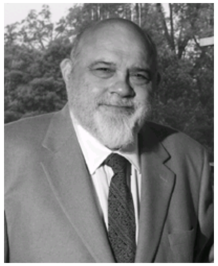
- u = 0 und v = 01 sind L-trennbar mit Hilfe von w = 0, denn $00 \notin L$ aber $010 \in L$
- 01 und 011 sind nicht L-trennbar.

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

Nerode-Lemma (1958)



Anil Nerode
*1932

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

- Beweis:

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

  $\Rightarrow \delta^*(q_0,w_1), \ldots, \delta^*(q_0,w_n)$ 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.

w v

u

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

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.

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 ε**ε** $\notin L_{drei}$ aber 0**ε** = 0 $\in L_{drei}$
  - 1 trennt ε und 1, denn ε1 $\notin L_{drei}$ aber 11 $\in L_{drei}$
  - 01 trennt ε und 10, denn ε01 $\notin L_{drei}$ aber 1001 $\in L_{drei}$
  - 0 trennt 0 und 1, denn 00 $\in L_{drei}$ aber 10 $\notin L_{drei}$
  - 0 trennt 0 und 10, denn 00 $\in L_{drei}$ aber 100 $\notin L_{drei}$
  - 1 trennt 1 und 10, denn 11 $\in L_{drei}$ aber 101 $\notin L_{drei}$

w v

u

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

> 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.

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 $\varepsilon\varepsilon \notin L_{drei}$ aber $0\varepsilon = 0 \in L_{drei}$
    - 1 trennt ε und 1, denn $\varepsilon 1 \notin L_{drei}$ aber $11 \in L_{drei}$
    - 01 trennt ε und 10, denn $\varepsilon 01 \notin L_{drei}$ aber $1001 \in L_{drei}$
    - 0 trennt 0 und 1, denn $00 \in L_{drei}$ aber $10 \notin L_{drei}$
    - 0 trennt 0 und 10, denn $00 \in L_{drei}$ aber $100 \notin L_{drei}$
    - 1 trennt 1 und 10, denn $11 \in L_{drei}$ aber $101 \notin L_{drei}$
  - Folglich hat jeder Automat, der L_{drei} erkennt mindestens 4 Zustände

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

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.

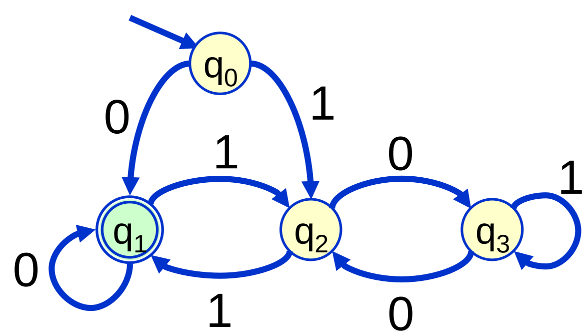
Anwendungen des Nerode-Lemmas

- 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 $\varepsilon\varepsilon \notin L_{drei}$ aber $0\varepsilon = 0 \in L_{drei}$
    - 1 trennt ε und 1, denn $\varepsilon 1 \notin L_{drei}$ aber $11 \in L_{drei}$
    - 01 trennt ε und 10, denn $\varepsilon 01 \notin L_{drei}$ aber $1001 \in L_{drei}$
    - 0 trennt 0 und 1, denn $00 \in L_{drei}$ aber $10 \notin L_{drei}$
    - 0 trennt 0 und 10, denn $00 \in L_{drei}$ aber $100 \notin L_{drei}$
    - 1 trennt 1 und 10, denn $11 \in L_{drei}$ aber $101 \notin L_{drei}$

  – Folglich hat jeder Automat, der L_{drei} erkennt mindestens 4 Zustände

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



Philipps Universität Marburg

Nerode-Lemma

- L_{bin} = { 0, 1, 10, 11, 101, … } = alle Binärzahlen ohne führende Nullen
  - –

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}

trennt	ε	0	1	01
ε				
0				
1				
01				

Nerode-Lemma

- $L_{bin} = \{ 0, 1, 10, 11, 101, \dots \}$ = alle Binärzahlen ohne führende Nullen
  - Eine trennbare Menge mit vier Elementen ist z.B.: $\{\varepsilon, 0, 1, 01\}$
    - ε trennt ε und 0, denn $\varepsilon\varepsilon \notin L_{bin}$ aber $0\varepsilon \in L_{bin}$
    - ε trennt ε und 1, denn $\varepsilon\varepsilon \notin L_{bin}$ aber $1\varepsilon \in L_{bin}$
    - 0 trennt ε und 01, denn $\varepsilon 0 \in L_{bin}$ aber $010 \notin L_{bin}$
    - ε trennt 0 und 01, denn $0\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - ε trennt 1 und 01, denn $1\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - 0 trennt 0 und 1, denn $00 \notin L_{bin}$ aber $10 \in L_{bin}$
  –

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

Nerode-Lemma

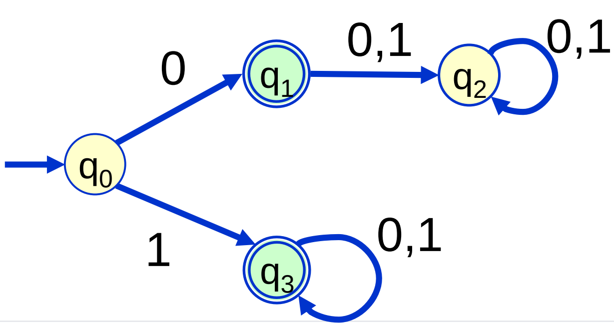
- $L_{bin} = \{ 0, 1, 10, 11, 101, \dots \}$ = alle Binärzahlen ohne führende Nullen
  - Eine trennbare Menge mit vier Elementen ist z.B.: $\{\varepsilon, 0, 1, 01\}$
    - ε trennt ε und 0, denn $\varepsilon\varepsilon \notin L_{bin}$ aber $0\varepsilon \in L_{bin}$
    - ε trennt ε und 1, denn $\varepsilon\varepsilon \notin L_{bin}$ aber $1\varepsilon \in L_{bin}$
    - 0 trennt ε und 01, denn $\varepsilon 0 \in L_{bin}$ aber $010 \notin L_{bin}$
    - ε trennt 0 und 01, denn $0\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - ε trennt 1 und 01, denn $1\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - 0 trennt 0 und 1, denn $00 \notin L_{bin}$ aber $10 \in L_{bin}$
  - 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				

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 $\varepsilon\varepsilon \notin L_{bin}$ aber $0\varepsilon \in L_{bin}$
    - ε trennt ε und 1, denn $\varepsilon\varepsilon \notin L_{bin}$ aber $1\varepsilon \in L_{bin}$
    - 0 trennt ε und 01, denn $\varepsilon 0 \in L_{bin}$ aber $010 \notin L_{bin}$
    - ε trennt 0 und 01, denn $0\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - ε trennt 1 und 01, denn $1\varepsilon \in L_{bin}$ aber $01\varepsilon \notin L_{bin}$
    - 0 trennt 0 und 1, denn $00 \notin L_{bin}$ aber $10 \in L_{bin}$
  - Folglich hat jeder Automat, der L_{bin} erkennt, mindestens 4 Zustände

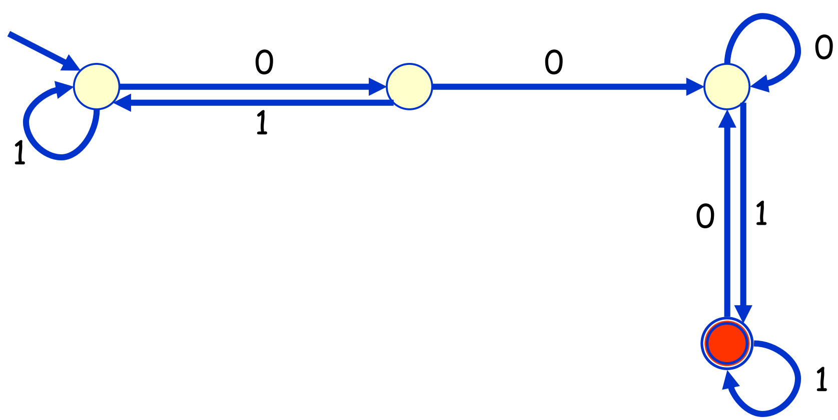
trennt	ε	0	1	01
ε		ε	ε	0
0			0	ε
1				ε
01				



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 1



Einfachster Automat für $L(A) \cap L(B)$?

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 1
- Zeige, dass z.B. $\{\varepsilon, 0, 00, 001\}$ eine trennbare Menge für L ist:
  - Dann muss jeder Automat für L mindestens 4 Zustände haben

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 1
- Zeige, dass z.B. $\{\varepsilon, 0, 00, 001\}$ eine trennbare Menge für L ist:
  - Dann muss jeder Automat für L mindestens 4 Zustände haben

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

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 ?

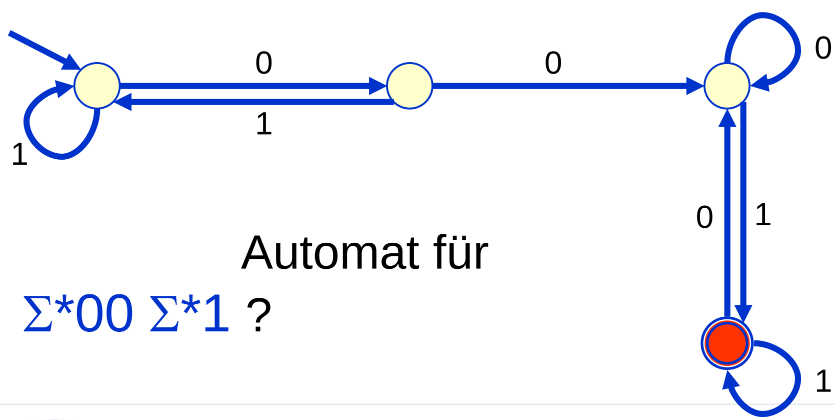
trennt	ε	0	00	001
ε		01	1	ε
0			1	ε
00				ε
001				

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:

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



Automat für $\Sigma^*00\,\Sigma^*1$?

Produktautomat

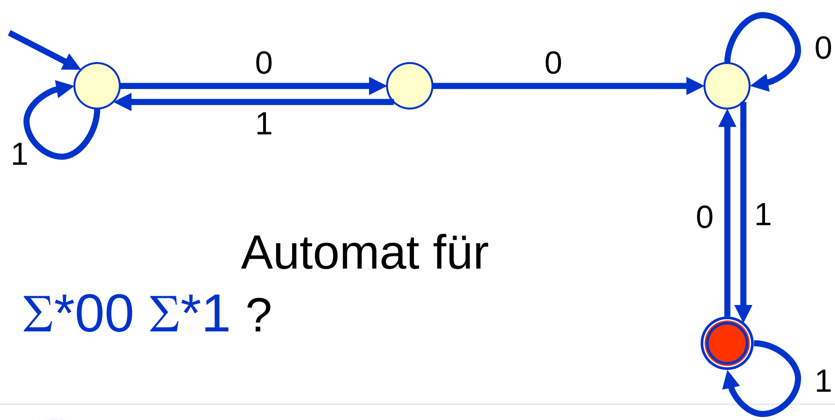
Produktkonstruktion lieferte nicht unbedingt den einfachsten Automat

- 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:
    $M=\{m_1,\dots,m_k\}$ trennbare Menge für L(A)
    $\Leftrightarrow$ $\{\ \delta^*(q_0,m_1), \dots , \delta^*(q_0,m_k)\}$ paarweise unterscheidbar

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



Automat für $\Sigma^*00\ \Sigma^*1$?

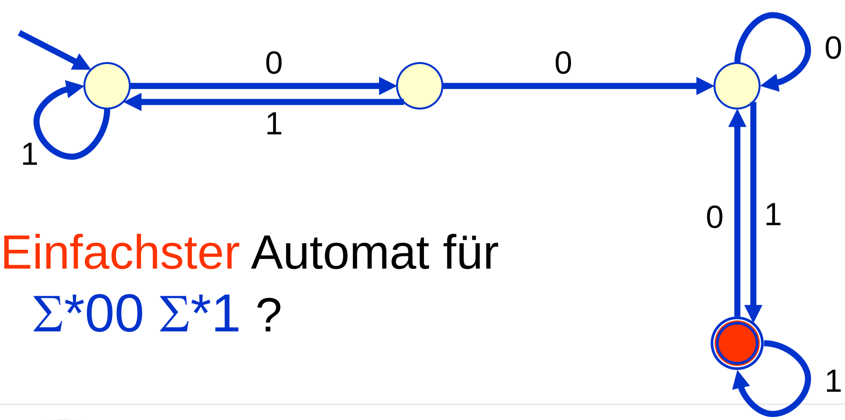
Philipps Universität Marburg

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) $\notin L_{bin}$ aber () $\in L_{bin}$
  -)) trennt ε und ((, denn ε)) $\notin L_{bin}$ aber (()) $\in L_{bin}$
  - …

  für $n \neq k$:
  - $)^n$ trennt $(^n$ und $(^k$, denn $(^n)^n \in L_{bin}$ aber $(^k)^n \notin L_{bin}$

  – 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		)	))	)))	))))
(			))	)))	))))
((				)))	))))
(((					))))
((((					

…

…

Philipps Universität Marburg

Palindrome nicht endlich erkennbar

- $L_{pal} = \{ w^R w \mid w \in \{a,b,c\}^* \}$ = Menge aller geraden Palindrome über $\Sigma=\{a,b,c\}$

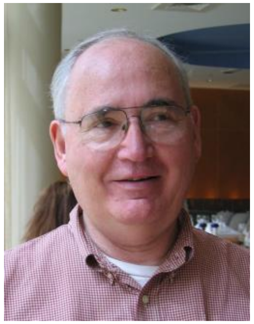
- Eine trennbare Menge ist z.B.: $\{ a^n b \mid n \geq 0 \}$, denn
  - für $k > i \geq 0$ gilt:

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

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

Pumping Lemma (1959)

Michael Rabin
*1931

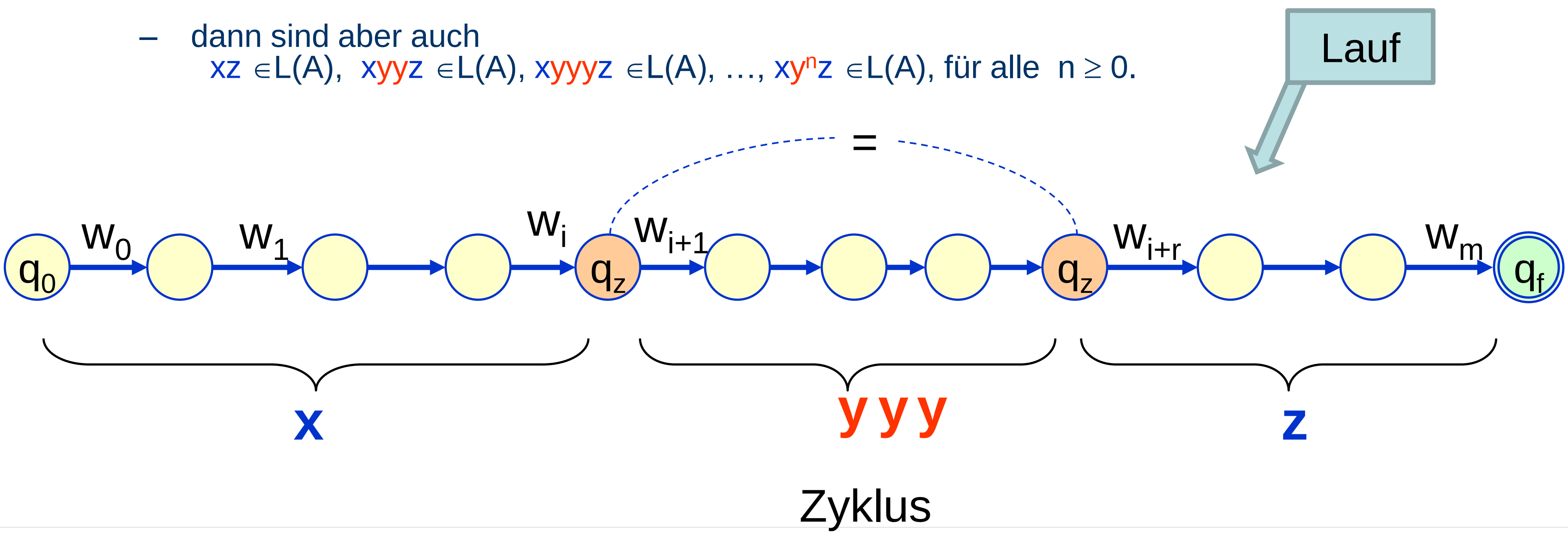


Dana Scott
*1932

- Alternatives Kriterium um zu zeigen, dass es keinen DFA für eine Sprache gibt.
- Komplizierter anzuwenden als Nerode aber beliebter in Textbüchern Schönes Konzept: Wortteile „aufpumpen“ oder „abpumpen“
- Idee: Wenn ein endlicher Automat mit $L = L(A)$ existiert mit $|A| = n$, dann muss jedes Wort $w \in L$ mit $|w| \geq n$ einen Zyklus durchlaufen.
- Diesen kann man dann aber auch mehrmals durchlaufen und erhält so immer größere Worte einer bestimmten Bauart in L…

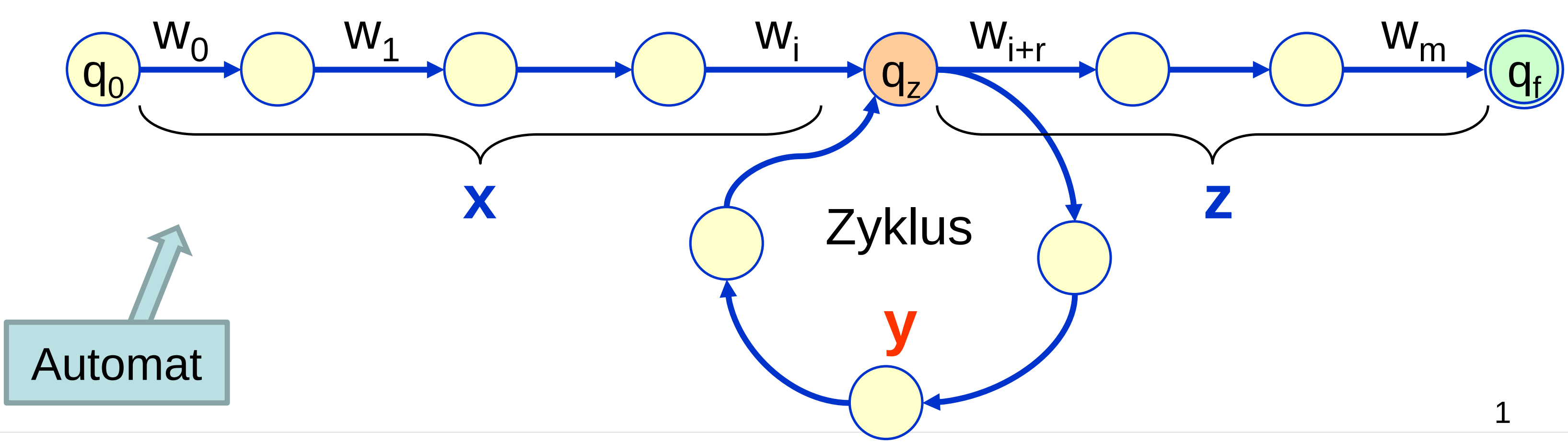
Lange Worte in kleinen Automaten

- 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 $n \geq 0$.



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 $n \geq 0$.



1

Pumping Lemma

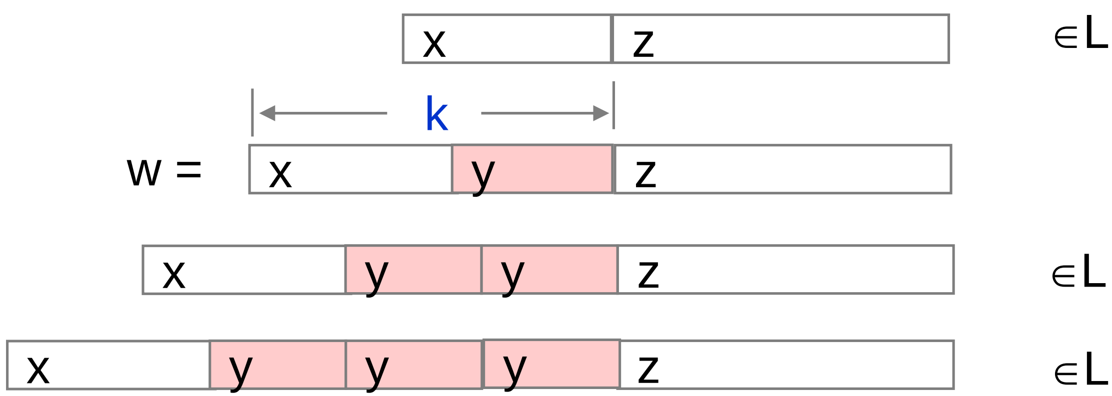
Für die Sprache L eines endlichen Automaten gibt es eine Zahl k, so dass jedes Wort $w \in L$ mit $|w| \geq k$ sich zerlegen lässt als

$$w = x\ y\ z$$

so dass

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

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



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.
  - Mit dem k von oben betrachten wir 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 |

$\underbrace{aaaaaaaaaaa}_{k}\underbrace{bbbbbbbbbbb}_{k} \in L$

| x | y y | z |

aber aaaaaaaaaaaaaabbbbbbbbbbb $\notin$ L

zu viele a-s

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

Logische Struktur des Pumping Lemma

L regulär $\Rightarrow$

$\exists k \in \mathbb{N}.$

$\forall w \in L, |w| \geq k$.

$\exists$ Zerlegung $w = xyz, |xy| \leq k, |y| \geq 1.$

$\forall n \geq 0. \, xy^n z \in L$

Umkehrung (zum Nachweis, dass Sprache nicht regulär ist):

$\forall k \in \mathbb{N}.$

$\exists w \in L, |w| \geq k$.

$\forall$ Zerlegung $w = xyz, |xy| \leq k, |y| \geq 1.$

$\exists n \geq 0. \, xy^n z \notin L$

$\Rightarrow$ L ist nicht regulär

Schema der Anwendung des Pumping Lemma

- Pumping Lemma (PL) ist **notwendig**, aber **nicht hinreichend** für Regularität!

- Behauptung: L ist nicht regulär.
- Beweis: Angenommen, L sei regulär, dann gilt das PL für L.
  - Habe k die Eigenschaft aus dem PL.
  - Dann wähle das spezielle Wort ... ∈ L mit | ... | ≥ k.
  - Betrachte eine beliebige Zerlegung von ... in xyz mit $|xy| \leq k$ und $|y| \geq 1$.
  - ... Konstruktion von neuen Wörtern aus L.
  - ⇒ Widerspruch, d.h. diese Wörter genügen nicht mehr den Eigenschaften der Sprache L.

Beispiel

- $L_{fact} = \{a^{m!} \mid m \geq 3\}$ ist Sprache keines endl. Automaten
  - Wähle ein k,
    - wähle $w = a^{k!}$ (garantiert $|w| \geq k$)
    - zerlege $w = a^{k!}$ als uzv mit $|uz| \leq k$, $|z| \geq 1$
    - 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 $L_{fact}=L(A)$

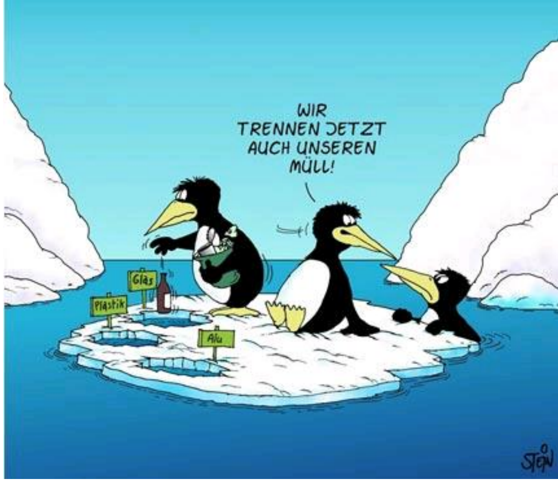
Beispiel

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

$$L_{ambm} = (\Sigma^* - L_{diff}) \cap a^*b^* \qquad L_{ambm} = \{a^m b^m \mid m \geq 0\}$$

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

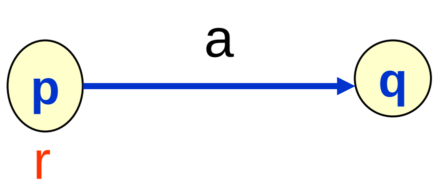
- $L_{ambm} = \{ a^m b^m \mid m \geq 0 \}$
  - $\{ a^i \mid i \geq 0 \}$ ist unendliche trennbare Menge
  - trenne a^i von allen a^j $(i \neq j)$ mittels b^i

- $L_{fact} = \{ a^{m!} \mid m \geq 3 \}$
  - $\{ a^{n!} \mid n \geq 0 \}$ ist unendliche trennbare Menge
  - falls $n < m$, trenne $a^{n!}$ von $a^{m!}$ mittels $a^{n! \cdot n}$:
    - $a^{n!}a^{n! \cdot n} = a^{(n+1)!} \in L_{fact}$, aber $a^{m!}a^{n! \cdot n} < a^{(m+1)!}$, daher $a^{m!}a^{n! \cdot n} \notin L_{fact}$,

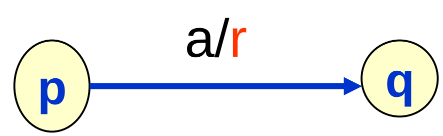
Es gibt daher keine endlichen Automaten A mit
$L_{fact}=L(A)$, $L_{ambm}=L(A)$, $L_{diff}=L(A)$.

Automaten mit Ausgabe

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



$\delta(p,a) = q$
$\gamma(p) = r$

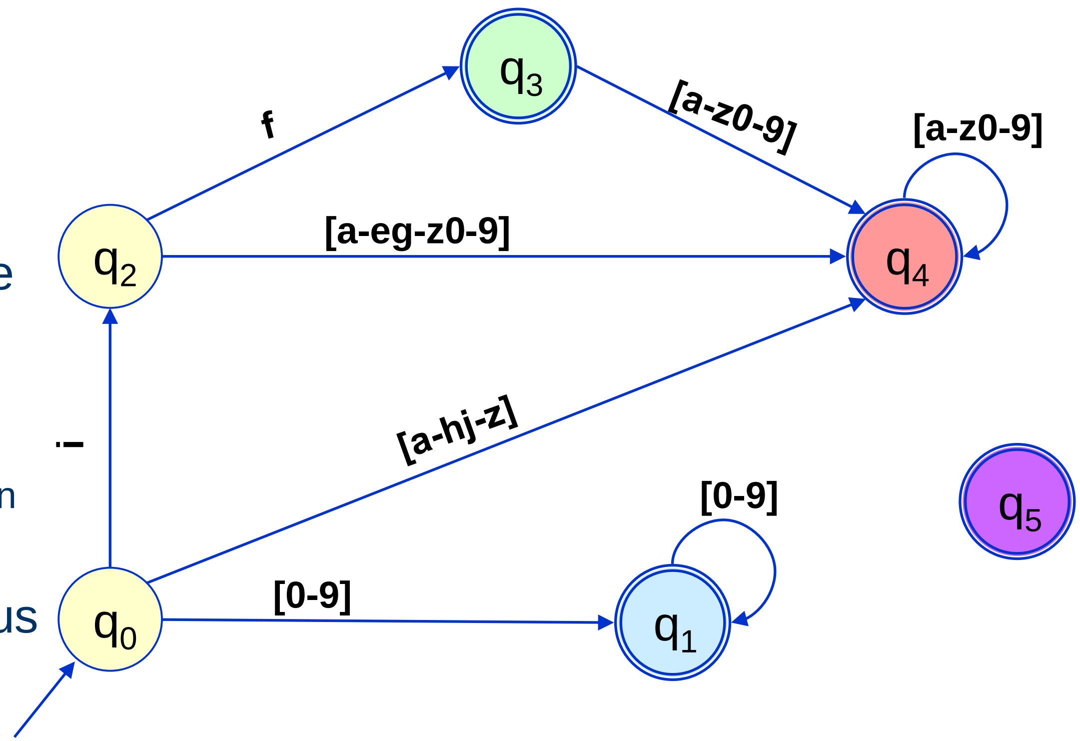


$\delta(p,a) = q$
$\gamma(p,a) = r$

- 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
  - Moore Automat
    - Ausgabe von Token in entsprechenden Endzuständen
  - Ausgabe-Token hält fest, aus welcher Sprache das erkannte Wort ist
    - Versuche, möglichst langes Präfix zu erkennen
    - dann wird abgeschnitten

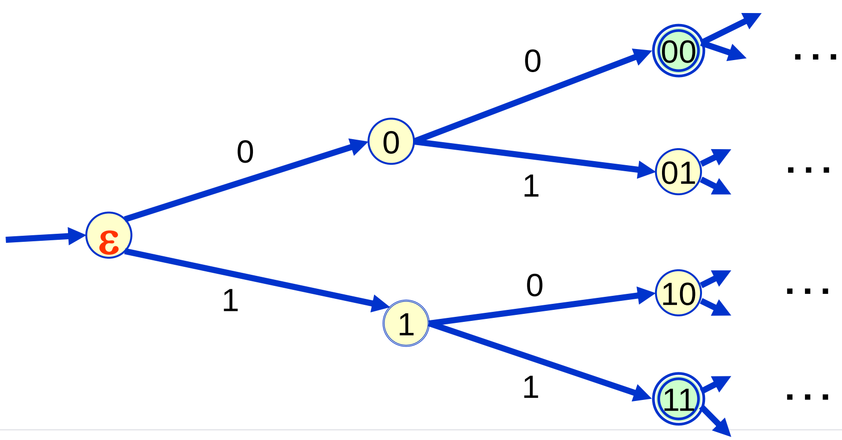


$q_3 \Rightarrow$ *ifToken*
$q_4 \Rightarrow$ *identifier*
$q_1 \Rightarrow$ *num*

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

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 Gerade Palindrome

Zusammenfassung DFA

- Einfaches Berechnungskonzept
  - Tabellengesteuerte Implementierung
  - Spracherkennung
  - L(A)
- Konstruktionen
  - Komplementautomat
  - Produktautomat
  - Homomorphismus
  - Minimalautomat
- Grenzen
  - Pumping Lemma
  - Trennbarkeit
  - Nerode Lemma
- Anwendung
  - Automaten mit Ausgabe
  - Scanner

Satz von Myhill und Nerode

$u\ R_L\ v \quad \text{gdw.} \quad \forall w \in \Sigma^*:$
$uw \in L \Leftrightarrow vw \in L$

R_L:

Nerode-Kongruenz oder

Nerode-Rechtskongruenz

- Einfach: Es gibt kein Wort w, mit welchem u, v in L getrennt werden können.

→ u, v liegen in der gleichen Äquivalenzklasse gemäß L-Trennbarkeit

- Relation R_L beschreibt die Äquivalenzklassen.
- $index(R_L)$: „Index" der Sprache L (Anzahl der Äquivalenzklassen)

→ Reguläre (durch Automaten erkennbare) Sprachen haben einen **endlichen** Index

Satz von Myhill und Nerode

$$u\ R_L\ v \quad \text{gdw.} \quad \forall w \in \Sigma^*: uw \in L \Leftrightarrow vw \in L$$

Satz (Myhill-Nerode):

Eine Sprache L ist genau dann regulär, wenn $index(R_L)$ endlich ist.

Beweis (Skizze, Details → Whiteboard):

⇒: Nehme DFA $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Definiere Äquivalenzrelation R_A auf Σ^*: $u\ R_A v\ :\Leftrightarrow\ \delta^*(q_0, u) = \delta^*(q_0, v)$. Zeige $index(R_L) \leq index(R_A)$.

⇐: Konstruiere DFA für L mit genau den R_L-Äquivalenzklassen als Zuständen (äquivalent zu Minimalautomat).