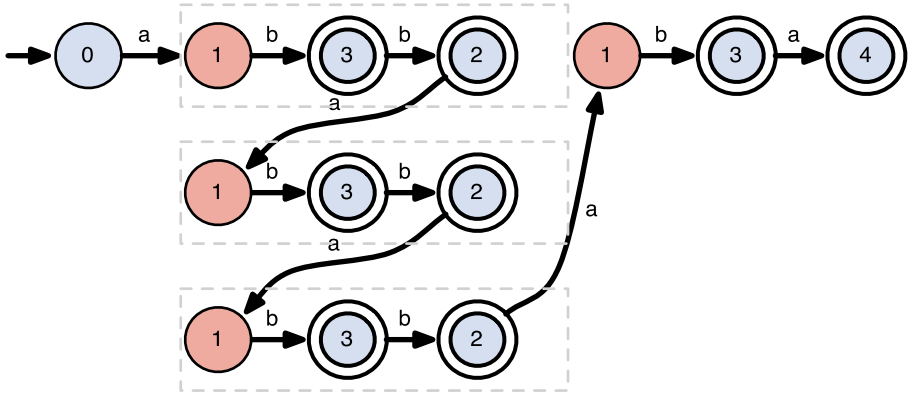
Berechenbarkeitstheorie

4. Vorlesung



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

Reguläre Ausdrücke

- Formalismus zum Beschreiben von formalen Sprachen
- ein Regulärer Ausdruck (RA) ist ein Wort über dem Alphabet $\Sigma \cup \{\emptyset, \varepsilon, (,), +, \cdot, *\}$

Syntax

(1) $\forall a \in \Sigma \cup \{\emptyset, \varepsilon\}$: a ist RA
(2) Wenn A, B Reg. Ausdrücke, dann auch $(A + B)$
(3) Wenn A, B Reg. Ausdrücke, dann auch $A \cdot B$ (kurz AB)
(4) Wenn A Reg. Ausdruck, dann auch A^*
(5) Wenn A Reg. Ausdruck, dann auch (A)

Bsp

$((\mathtt{aba}^* + \mathtt{abb})^*\mathtt{ab} + \varepsilon)$ ist RA

$(\mathtt{a} + \mathtt{b})\mathtt{b}(^*\mathtt{b})$ ist kein RA

Induktive Definition korrekt, da RA durch die Anwendung der (umgekehrten) Regeln immer kürzer werden

Semantik Regulärer Ausdrücke

Für jeden RA R definieren wir induktiv eine Sprache $L(R)$:

(1)
- $R = \emptyset$, dann $L(R) = \emptyset$
- $R = \varepsilon$, dann $L(R) = \{\varepsilon\}$
- $R = \mathsf{a}$, und $\mathsf{a} \in \Sigma$, dann $L(R) = \{\mathsf{a}\}$

(2) $R = (A + B)$ und A, B sind RA, dann $L(R) = L(A) \cup L(B)$

(3) $R = (AB)$ und A, B sind RA, dann $L(R) = L(A) \circ L(B)$

(4) $R = A^*$ und A ist RA, dann $L(R) = L(A)^*$

(5) $R = (A)$ und A ist RA, dann $L(R) = L(A)$

Reihenfolge der Operatoren: $*$ vor $\cdot$ vor $+$

Bsp. $R = (\mathsf{a} + \mathsf{b})^*\mathsf{aba}(\mathsf{a} + \mathsf{b})^*$

$L(R) = \{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid w \text{ enthält Teilwort } \mathsf{aba}\}$

Satz von Kleene

Satz 5

$$\{L \mid \exists \text{ RA } R \text{ mit } L(R) = L\} = \text{REG}$$

Beweis (Organisation)

1. Teil: Wir konstruieren zu einem RA R einen NEA N mit $L(R) = L(N)$

das zeigt $\{L \mid \exists \text{ RA } R \text{ mit } L(R) = L\} \subseteq \text{REG}$

2. Teil: Wir konstruieren zu einem DEA M einen RA R mit $L(M) = L(R)$

heute

das zeigt $\{L \mid \exists \text{ RA } R \text{ mit } L(R) = L\} \supseteq \text{REG}$

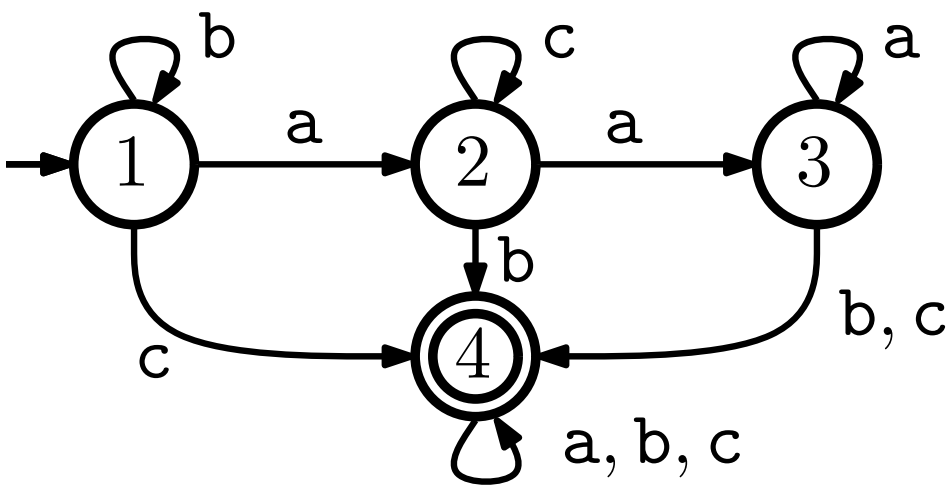
Vom DEA zum RA

Ansatz

- Wenn $L \in$ REG, dann existiert ein DEA M der L erkennt.
- $M = (Q, \Sigma, \delta, 1, F)$ mit $Q = \{1, 2, \ldots, n\}$
- **Ziel:** Konstriere RA R mit $L(R) = L$

$R_{ij}^k :=$ RA für alle Wörter die von Zustand i nach j führen ohne einen Zustand $> k$ zu benutzen.

Anfangs- und Endzustand i, j dürfen $> k$ sein!



Bsp $R_{22}^0 = \mathtt{c} + \varepsilon$

$R_{24}^2 = \mathtt{c}^*\mathtt{b}$

$R_{24}^3 = (\mathtt{c}^*\mathtt{b} + \mathtt{c}^*\mathtt{a}\mathtt{a}^*(\mathtt{b} + \mathtt{c}))$

Von den Termen zum RA

Annahme: Ich kenne alle Terme R_{ij}^k

$\rightarrow w \in L(M)$ genau dann, wenn $\underbrace{\delta^*(1, w) = j}_{w \in R_{1j}^n}$ und $j \in F$

oberer Index n heißt keine Beschränkung

$\rightarrow$ verbinde für alle akzeptierende Zustände die R_{ij}^k Terme

$$R := R_{1f_1}^n + R_{1f_2}^n + \cdots R_{1f_k}^n$$

wobei $F = \{f_1, f_2, \ldots, f_k\}$

Konstruktion der R_{ij}^k Terme

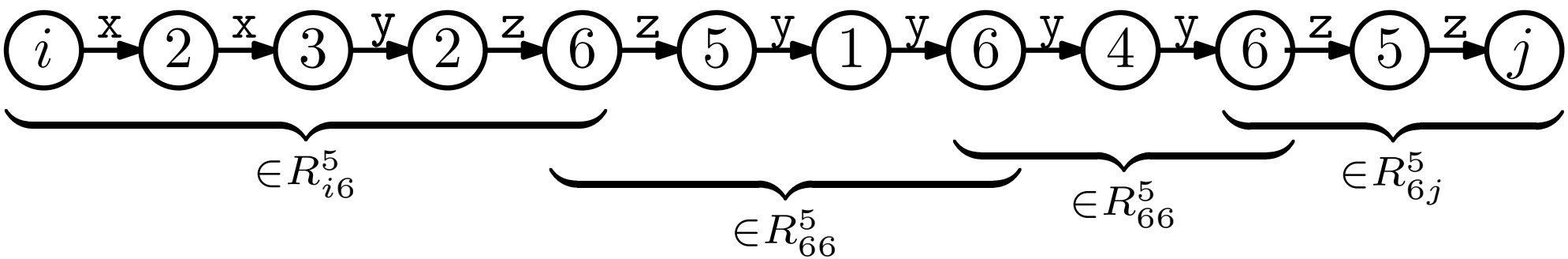
Rekursive Konstuktion

Basisfall (R_{ij}^0): nur direkte Übergänge

$$R_{ij}^0 := \begin{cases} \emptyset + a_1 + a_2 + \dots & \text{falls } i \neq j \text{ und } a_i \in \{a \mid \delta(i,a) = j\} \\ \varepsilon + a_1 + a_2 + \dots & \text{falls } i = j \text{ und } a_i \in \{a \mid \delta(i,a) = i\} \end{cases}$$

Rekursion (R_{ij}^k): ich kenne bereits alle R_{ij}^{k-1}

Bsp. von einem "Lauf" aus R_{ij}^6



$\rightarrow$ R_{**}^k **kann ich mit Termen** R_{**}^{k-1} **beschreiben**

Rekursion für R_{ij}^k

- 1. Möglichkeit: der *Lauf* eines Wortes besucht **nicht** den Zustand k

  $\rightarrow R_{ij}^k \supseteq R_{ij}^{k-1}$

- 2. Möglichkeit: der *Lauf* eines Wortes besucht einmal oder mehrmals den Zustand k

  $\rightarrow R_{ij}^k \supseteq R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

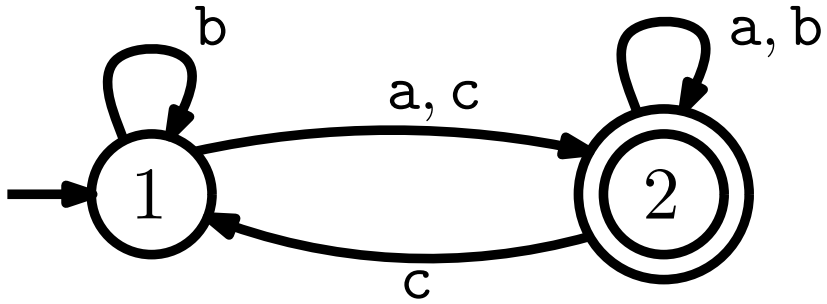
- es gibt keine andere Möglichkeit, deshalb

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$$

Berechnung mit ansteigendem k $(k = 0, \ldots, n)$ aller R_{ij}^k Terme

Dynamisches Programmieren

Beispiel



$$R_{ij}^0 := \begin{cases} \emptyset + a_1 + a_2 + \ldots & \text{falls } i \neq j \text{ und } a_i \in \{a \mid \delta(i,a) = j\} \\ \varepsilon + a_1 + a_2 + \ldots & \text{falls } i = j \text{ und } a_i \in \{a \mid \delta(i,a) = i\} \end{cases}$$

$$R_{11}^0 = \mathtt{b} + \varepsilon \quad R_{12}^0 = \mathtt{a} + \mathtt{c} \quad R_{21}^0 = \mathtt{c} \quad R_{22}^0 = (\mathtt{a} + \mathtt{b} + \varepsilon)$$

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$$

$$\begin{aligned} R_{11}^1 &= (\mathtt{b} + \varepsilon) + (\mathtt{b} + \varepsilon)(\mathtt{b} + \varepsilon)^*(\mathtt{b} + \varepsilon) = \mathtt{b}^* \\ R_{12}^1 &= (\mathtt{a} + \mathtt{c}) + (\mathtt{b} + \varepsilon)(\mathtt{b} + \varepsilon)^*(\mathtt{a} + \mathtt{c}) = \mathtt{b}^*(\mathtt{a} + \mathtt{c}) \\ R_{21}^1 &= \mathtt{c} + \mathtt{c}(\mathtt{b} + \varepsilon)^*(\mathtt{b} + \varepsilon) = \mathtt{c}\mathtt{b}^* \\ R_{22}^1 &= (\mathtt{a} + \mathtt{b} + \varepsilon) + \mathtt{c}(\mathtt{b} + \varepsilon)^*(\mathtt{a} + \mathtt{c}) = (\mathtt{a} + \mathtt{b} + \varepsilon) + \mathtt{c}\mathtt{b}^*(\mathtt{a} + \mathtt{c}) \end{aligned}$$

$$R = R_{12}^2 =$$

$$\mathtt{b}^*(\mathtt{a} + \mathtt{c}) + \mathtt{b}^*(\mathtt{a} + \mathtt{c})\left[(\mathtt{a} + \mathtt{b} + \varepsilon) + \mathtt{c}\mathtt{b}^*(\mathtt{a} + \mathtt{c})\right]^* \left[(\mathtt{a} + \mathtt{b} + \varepsilon) + \mathtt{c}\mathtt{b}^*(\mathtt{a} + \mathtt{c})\right]$$

Fazit

Reguläre Sprachen

vom **DEA** erkannt (nach Def.)

Satz von Kleene

Potenzautomat

äquivalent zu **RA**

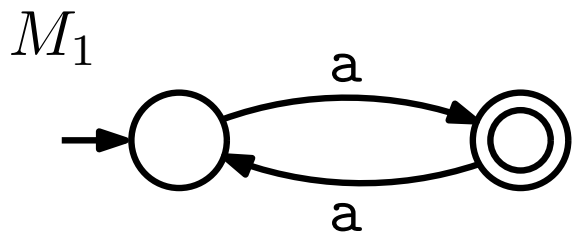
Formalismus zum Beschreiben formaler Sprachen

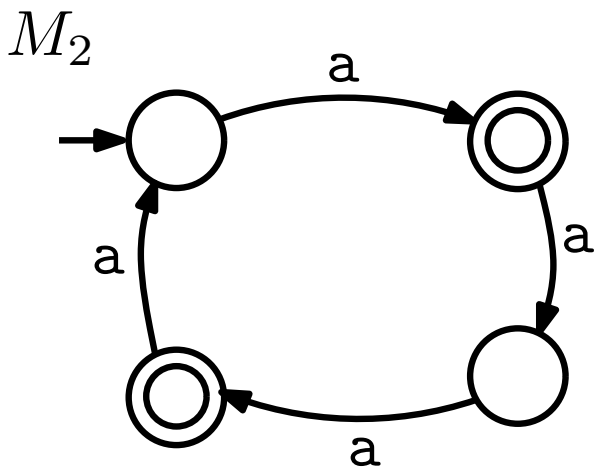
äquivalent zum **NEA**

Berechnungsmodell mit

- Nichtdeterminismus
- ε-Übergängen

Minimierung von DEAs





$L(M_1) = \{\mathsf{a}^k \mid k \text{ ist ungerade}\}$ $L(M_2) = \{\mathsf{a}^k \mid k \text{ ist ungerade}\}$

Definition

Zwei DEAs M_1 und M_2 heißen **äquivalent**, gdw.

$$L(M_1) = L(M_2)$$

Frage: Welches ist der kleinste DEA für eine Sprache L?

minimale Anzahl von Zuständen

Äquivalenz von Zuständen

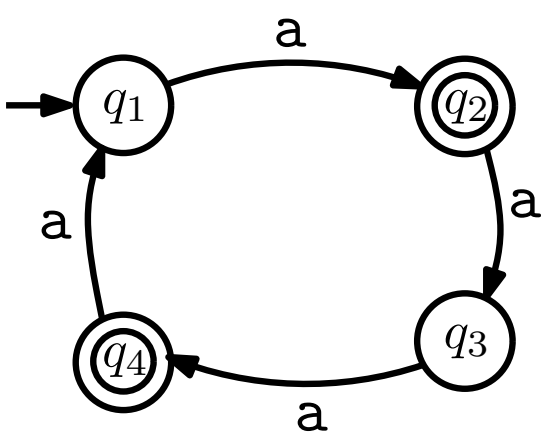
Definition

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA, dann heißen zwei Zustände $p, q \in Q$ **äquivalent** (Schreibweise $p \approx q$) gdw.

$$\forall z \in \Sigma^*: \quad \delta^*(p, z) \in F \Leftrightarrow \delta^*(q, z) \in F$$

Zwei nicht äquivalente Zustände nennen wir auch trennbar.

Bsp.



- q_1 und q_2 sind trennbar (z.B. durch das Trennwort a)
- q_1 und q_3 sind äquivalent

Lemma 2

Die Relation $\approx$ ist eine Äquivalenzrelation.

Erinnerung: $p \approx q$ wenn $\forall z \in \Sigma^*$: $\quad \delta^*(p,z) \in F \Leftrightarrow \delta^*(q,z) \in F$

Symmetrie

$$\delta^*(p,z) \in F \Leftrightarrow \delta^*(q,z) \in F$$

$$\delta^*(q,z) \in F \Leftrightarrow \delta^*(p,z) \in F$$

Reflexivität $\quad \forall p \in Q: \quad \delta^*(p,z) \in F \Leftrightarrow \delta^*(p,z) \in F$

Transitivität

$$\left[\begin{array}{l} \delta^*(p,z) \in F \Leftrightarrow \delta^*(q,z) \in F \\ \delta^*(q,z) \in F \Leftrightarrow \delta^*(r,z) \in F \end{array}\right] \Rightarrow \delta^*(p,z) \in F \Leftrightarrow \delta^*(r,z) \in F$$

□

$\rightarrow$ die Zustandsmenge Q zerfällt in Äquivalenzklassen

$\rightarrow$ Äquivalenzklasse die $q \in Q$ enthält $[q] := \{p \in Q \mid p \approx q\}$

Für zwei Zustände p, q aus einer Klasse gilt

1. $p \in F \Leftrightarrow q \in F$,
2. $\forall a \in \Sigma$, dass $\delta(p,a) \approx \delta(q,a)$.

Der kollabierte Automat

Definition

Der **kollabierte Automat** $M' = (Q', \Sigma, \delta', q_0', F')$ ist ein DEA mit

- $Q' = \{[q] \mid q \in Q\}$,
- $\delta'([q], a) = [\delta(q, a)]$,
- $q_0' = [q_0]$,
- $F' = \{[q] \mid q \in F\}$.

Außerdem werden alle nicht erreichbaren Zustände gestrichen.

Definition ist korrekt, denn

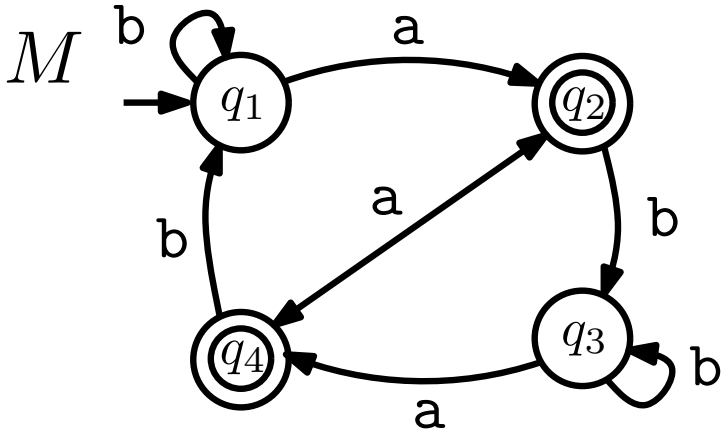
- Alle Elemente aus $[q]$ beschreiben den gleichen Folgezustand bzgl. δ'

$$p \approx q \quad \Rightarrow \quad \delta(p, \mathtt{a}) \approx \delta(q, \mathtt{a})$$

- Akzeptierte Klassen sind wohldefiniert

$$p_1, p_2 \in [q] \quad \Rightarrow \quad \delta(p_1, \varepsilon) \in F \Leftrightarrow \delta(p_2, \varepsilon) \in F$$
$$\Rightarrow \quad p_1 \in F \Leftrightarrow p_2 \in F$$

Beispiel: Der kollabierte Automat

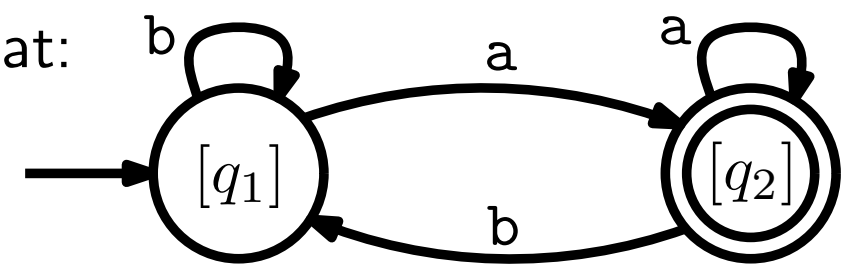


Alle Zustandspaare trennbar bis auf

- $q_1 \approx q_3$,
- $q_2 \approx q_4$.

- zwei Klassen: $[q_1] = \{q_1, q_3\}$ und $[q_2] = \{q_2, q_4\}$, d.h. $Q' = \{[q_1], [q_2]\}$
- nur $[q_2]$ enthält akz. Zustände, deshalb $F' = \{[q_2]\}$
- $[q_1]$ enthält Startzustand, deshalb $q_0' = [q_1]$

Kollabierter Automat:



Lemma 2

Sei M' der kollabierte Automat von M, dann gilt

$$L(M) = L(M').$$

Beweis

- wir betrachten ein $w \in \Sigma^*$, $w = x_1 x_2 \cdots x_k$, mit $x_i \in \Sigma$
- sei $(s_1, s_2, \ldots, s_{k+1})$ der w-Lauf in M $(s_i \in Q)$

  Zur Erinnerung
  1. $q_0 = s_1$,
  2. $\delta(s_i, x_i) = s_{i+1}$ für alle $1 \leq i \leq k$.
- $([s_1], [s_2], \ldots, [s_{k+1}])$ ist der w-Lauf in M', denn
  1. $q_0' = [q_0] = [s_1]$,
  2. $\delta'([s_i], x_i) = [\delta(s_i, x_i)] = [s_{i+1}]$ für alle $1 \leq i \leq k$.
- $[s_{k+1}] \in F' \iff s_{k+1} \in F$
- w-Lauf akzeptierend in M gdw. w-Lauf akzeptierend in M'
- $w \in L(M) \iff w \in L(M')$

□

Table-Filling Algorithmus

- Algorithmus zum effizienten Finden aller äquivalenten Zustände
- Datenstruktur: Tabelle T, mit $|Q|$ Zeilen und Spalten, $Q = \{1, 2, 3, \ldots, n\}$.
- **Invariante:** Enthält $T[p, q]$ die Markierung **1** dann sind p und q trennbar ($p \not\approx q$)
- Tabelle T wird nach und nach mit 1en gefüllt bis eine Abruchbedingung eintritt
- am Ende notieren alle unmarkierten Einträge $T[p, q]$ äquivalente Zustandspaare p, q

Table-Filling Algorithmus

- Wir füllen nur die Hälfte der Tabelle T aus ($T[p,q]$ mit $p < q$)

Algorithm 1: TableFilling Algorithmus

```
1 Initialisiere T ≡ 0;
2 for all (p, q) ∈ Q × Q do
3 |   if (p ∈ F und q ∉ F) oder (p ∉ F und q ∈ F) then
  |   T[p, q] = 1;
4 end
5 repeat
6 |   for all (p, q) ∈ Q × Q mit T[p, q] ≠ 1 do
7 |   |   if ∃a ∈ Σ: T[δ(p, a), δ(q, a)] == 1 then T[p, q] = 1;
8 |   end
9 until keine neue Markierung gesetzt;
```

Table-Filling Algorithmus

- Wir füllen nur die Hälfte der Tabelle T aus ($T[p,q]$ mit $p < q$)

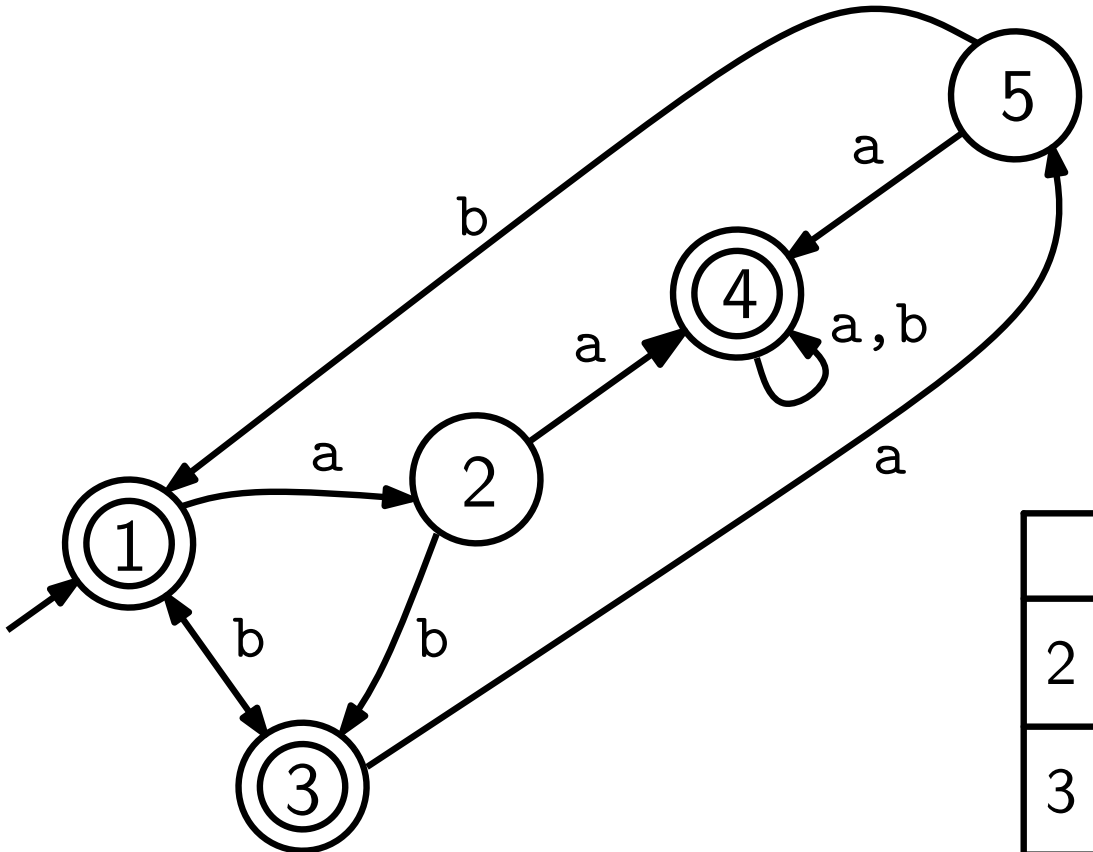
Algorithm 1: TableFilling Algorithmus

```
1 Initialisiere T ≡ 0;                                   Initialisierung
2 for all (p,q) ∈ Q × Q do
3 |   if (p ∈ F und q ∉ F) oder (p ∉ F und q ∈ F) then
  |   T[p,q] = 1;
4 end
5 repeat
6 |   for all (p,q) ∈ Q × Q mit T[p,q] ≠ 1 do
7 |   |   if ∃a ∈ Σ: T[δ(p,a), δ(q,a)] == 1 then T[p,q] = 1;
8 |   end                          Bedingung zum Setzen neuer Marken
9 until keine neue Markierung gesetzt;
```

Abbruchbedingung

Table-Filling Algorithmus Beispiel

Table-Filling Algorithmus Beispiel



	1	2	3	4
2	1			
3		1		
4	1	1	1	
5	1		1	1

Keine weiteren Veränderungen

$\rightarrow 1 \approx 3$ und $2 \approx 5$

Korrektheit TF-Algorithmus

Zu zeigen:
1. nur trennbare Paare wurden markiert (Invariante bleibt erhalten)
2. alle trennbaren Paare wurden markiert

1.

- 2 Möglichkeiten wie markiert wurde
- Bei der Initialisierung:
  if $(p \in F$ und $q \notin F)$ oder $(p \notin F$ und $q \in F)$
  then $T[p, q] = 1$

  $\rightarrow p, q$ trennbar mit ε
- In der **repeat** Schleife:
  if $\exists a \in \Sigma$: $T[\delta(p, a), \delta(q, a)] == 1$ **then** $T[p, q] = 1$

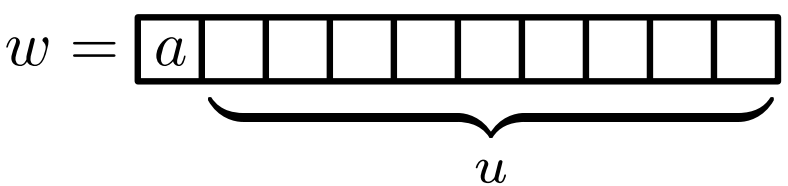
  $\rightarrow \delta(p, a), \delta(q, a)$ trennbar mit $w \in \Sigma^*$ (Invariante)

  $\rightarrow p, q$ trennbar mit aw

2. Alle trennbaren Paare wurden markiert

Schlechtes Paar: trennbar aber nicht markiert

- Annahme: sei (p, q) schlechtes Paar mit kürzestem Trennwort w und kein schlechtes Paar besitzt ein Trennwort kürzer als w
- $w \neq \varepsilon$, denn sonst wäre (p, q) während der Initialisierung markiert worden
- $w = au$ mit $a \in \Sigma$
- $p' = \delta(p, a)$ und $q' = \delta(q, a)$



OBdA

$$\delta^*(p', u) = \delta^*(p, w) \in F$$
$$\delta^*(q', u) = \delta^*(q, w) \notin F$$

- $T[p', q'] \neq 1$, denn sonst wäre während der Ausführung $T[p, q] = 1$ gesetzt
- **aber:** (p', q') trennbar mit u, und $|u| < |w|$

$\rightarrow$ Widerspruch zur Annahme, d.h. es gibt kein schlechtes Paar!