

Grammatiken

Das fundamentale Modell zur Beschreibung von formalen Sprachen durch Erzeugungsmechanismen sind Grammatiken.

Eine **Grammatik** ist ein Quadrupel (N, T, P, S) wobei

- N das Alphabet der **Nonterminale** (Variablen),
- T das Alphabet der **Terminalsymbole**,
- P eine Menge von **Produktionen**,
- $S \in N$ das **Startsymbol** ist.

Üblicherweise ist $N \cap T = \{\}$.

Wir definieren $V := N \cup T$.

$P \subseteq V^+ \times V^*$,

d.h., jede Produktion p aus P ist von der Gestalt $p = (\alpha, \beta)$ mit $\alpha \in V^+$ und $\beta \in V^*$.

Anstelle von (α, β) schreiben wir auch $\alpha \rightarrow \beta$.

Grammatiken: Ableitung

Sei $G = (N, T, P, S)$ eine Grammatik.

Ein Wort $w \in V^*$ heißt **ableitbar** in G aus dem Wort $v \in V^+$, in Symbolen $v \xRightarrow[G]{\text{pink}} w$, falls Wörter $x, y \in V^*$ derart existieren, dass $v = x\alpha y$ und $w = x\beta y$ für eine Produktion $(\alpha, \beta) \in P$ gilt.

Durch $\xRightarrow[G]{\text{pink}}$ wird eine Relation über V^* definiert.

$\xRightarrow[G]{*}$ Reflexive und transitive Hülle von $\xRightarrow[G]{\text{pink}}$

$\xRightarrow[G]{\text{pink}}$ Ableitung in einem Schritt

$\xRightarrow[G]{n}$ Ableitung in n Schritten

Ist G eindeutig aus dem Zusammenhang erkennbar, so schreiben wir \Rightarrow statt $\xRightarrow[G]{\text{pink}}$ etc.

Grammatiken: erzeugte Sprache

Sei $G = (N, T, P, S)$ eine Grammatik.

Gilt $S \xRightarrow[G]{\quad} w$ für ein Wort $w \in V^*$, so nennt man w **Satzform**.

Menge aller in n Schritten ableitbaren Satzformen:

$$SF(G, n) = \{ w \in V^* \mid S \xRightarrow[G]{n} w \}$$

Die **von G erzeugte Sprache** ist die Menge aller Wörter (Satzformen), die in beliebig vielen Schritten von S abgeleitet werden können und nur aus Terminalsymbolen bestehen:

$$L(G) = \{ w \in T^* \mid S \xRightarrow[G]{*} w \}$$

Grammatiken: Beispiel 1

Beispiel:

$$G_1 = (\{ S \}, \{ a \}, \{ S \rightarrow \varepsilon, S \rightarrow aS \}, S)$$

$$L(G_1) = \{ \varepsilon \} \cup \{ a^n \mid n \geq 1 \} = \{ a \}^*$$

Alle in G möglichen Ableitungen sind von der Gestalt

$$S \Rightarrow \varepsilon \text{ bzw. } S \Rightarrow aS \Rightarrow^n a^{n+1}S \Rightarrow a^{n+1}$$

für ein $n \in \mathbf{N}$.

Grammatiken: Beispiel 2

Beispiel:

$$G_2 = (\{S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S)$$

$$L(G_2) = \{ a^n b^n \mid n \in \mathbf{N} \}$$

Alle in G möglichen Ableitungen sind von der Gestalt

$$S \Rightarrow^n a^n S b^n \Rightarrow a^n b^n \quad \text{für alle } n \in \mathbf{N}.$$

Formaler Beweis mittels natürlicher Induktion:

Menge aller Satzformen nach genau n Schritten:

$$SF(G_2, n) = \{ a^n S b^n, a^{n-1} b^{n-1} \}$$

Grammatiken: Beispiel 2 (Induktion)

$$G_2 = (\{S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \epsilon \}, S)$$

Formaler Beweis mittels natürlicher Induktion.

Menge aller Satzformen nach genau $n \geq 1$ Schritten:

$$SF(G_2, n) = \{ a^n S b^n, a^{n-1} b^{n-1} \}$$

Induktionsbasis: $SF(G_2, 1) = \{ a^1 S b^1, \epsilon \}$

Induktionshypothese: $SF(G_2, n) = \{ a^n S b^n, a^{n-1} b^{n-1} \}$

Induktionsbehauptung: $SF(G_2, n+1) = \{ a^{n+1} S b^{n+1}, a^n b^n \}$

Beweis: Das Wort $a^{n-1} b^{n-1}$ ist terminal und daher nicht mehr weiter ableitbar. Aus $a^n S b^n$ ist ableitbar:

mittels $S \rightarrow aSb$: $a^{n+1} S b^{n+1}$

mittels $S \rightarrow \epsilon$: $a^n b^n$

□

Grammatiken: Beispiel 3

Beispiel:

$G_3 = (\{S,A,C\}, \{a,b,c\}, P_3, S)$ wobei

$P_3 = \{ S \rightarrow abc, S \rightarrow aAbc, A \rightarrow aAbC, A \rightarrow abC, \\ Cb \rightarrow bC, Cc \rightarrow cc \}$

$$L(G_3) = \{a^n b^n c^n \mid n \in \mathbf{N}_1 \}$$

Alle in G möglichen Ableitungen sind von der Gestalt

$$S \Rightarrow abc$$

bzw. für $n \geq 2$:

$$S \Rightarrow aAbc \Rightarrow^{n-2} a^{n-1}A(bC)^{n-2}bc \Rightarrow a^n (bC)^{n-1}bc \Rightarrow^* a^n b^n c^n$$

Grammatik-Typen

Die vorhergehenden Beispiele zeigen, dass zur Erzeugung bestimmter formaler Sprachen Produktionen mit wachsender Komplexität benötigt werden.

Aufgrund dieser Komplexität der Produktionen können wir verschiedene Typen von Grammatiken definieren.

Typ-i-Grammatiken

Sei $G=(N,T,P,S)$ eine Grammatik.

Dann heißt G auch **unbeschränkte Grammatik** (Typ-0)

Gilt für alle Produktionen $(\alpha,\beta) \in P$

- $|\alpha| \leq |\beta|$, so heißt G **monoton**;
- $\alpha = uAv$ und $\beta = uwv$ für ein $A \in N$, $w \in V^+$ und $u,v \in V^*$,
so heißt G **kontextsensitiv** (Typ-1)
(**monoton und kontextsensitiv**); außerdem:
kommt S nicht auf der rechten Seite einer Produktion vor,
so ist auch $S \rightarrow \varepsilon$ erlaubt;
- $A \rightarrow \beta$ für ein $A \in N$, so heißt G **kontextfrei** (Typ-2);
- $A \rightarrow aB$ oder $A \rightarrow \varepsilon$ für $A,B \in N$ und $a \in T$, so heißt G
regulär (Typ-3).

Erzeugte Sprachen

Eine formale Sprache heißt **rekursiv aufzählbar**, **monoton** (**kontextsensitiv**), **kontextfrei** bzw. **regulär**, wenn sie von einer Typ-0-, Typ-1-, Typ-2-, bzw. Typ-3-Grammatik erzeugt wird.

Aufgrund der Definition können wir nun die einzelnen Sprachen aus den vorigen Beispielen klassifizieren:

Es ergibt sich, dass

$L(G_1)$ regulär

$L(G_2)$ kontextfrei und

$L(G_3)$ monoton ist.

Zwei Grammatiken G und G' heißen **äquivalent**, wenn $L(G) = L(G')$.

Äquivalenz von kontextsensitiven und monotonen Grammatiken

Nach Definition ist jede kontextsensitive Grammatik auch eine monotone Grammatik.

Es gilt allerdings auch die Umkehrung:

Satz. Zu jeder monotonen Grammatik kann man eine äquivalente kontextsensitive Grammatik konstruieren.

Aufgabe MONA*: Beweisen Sie obigen Satz.

Grenzen der Berechenbarkeit (Grammatiken)

Die Menge aller formaler Sprachen $L \subseteq \Sigma^*$ ist überabzählbar, doch nur eine abzählbare Menge davon ist von einer Grammatik erzeugbar. (Kodierung der Beschreibungen von Grammatiken über 0,1).

Die Menge aller formalen Sprachen $L \subseteq \Sigma^*$, die von einer Grammatik erzeugt werden können, ist abzählbar.

Gibt es aber vielleicht andere Modelle von Generierungs- oder Analysemechanismen, durch die mehr formale Sprachen als durch Typ-0-Grammatiken beschrieben werden können?

Grenzen der Berechenbarkeit



1936 erfanden gleichzeitig **Alan Turing** die Turingmaschinen und **Alonzo Church** den λ -Kalkül, um den Begriff des Algorithmus bzw. der berechenbaren Funktionen zu formalisieren, und beide Modelle erwiesen sich als gleichwertig. Auch alle anderen seither entwickelten Modelle zur Formalisierung des Begriffs Algorithmus erwiesen sich als nicht mächtiger als Typ-0-Grammatiken bzw. Turingmaschinen. Die folgende These wird daher allgemein akzeptiert:

These von Turing

Gibt es ein endlich beschreibbares Verfahren zur exakten Spezifizierung einer formalen Sprache L , so gibt es eine Typ-0-Grammatik, die L erzeugt bzw. eine Turingmaschine, die L akzeptiert.

Sprachfamilien

Jeder der vorgestellten Grammatiktypen definiert auch eine *Familie formaler Sprachen*:

Sei $i \in \{0,1,2,3\}$ und Σ ein Alphabet.

Dann wird die Menge aller formaler Sprachen $L \subseteq \Sigma^*$, die von einer Grammatik vom Typ i erzeugt werden können, mit $L_i(\Sigma)$ bezeichnet. Die **Familie der formalen Sprachen**, die von einer Typ- i -Grammatik erzeugt werden können, bezeichnen wir mit **L_i** .

Sprachfamilien (rekursive Sprachen)

Sei Σ ein Alphabet. Eine formale Sprache $L \subseteq \Sigma^*$ heißt genau dann **rekursiv**, wenn sowohl $L \in \mathbf{L}_0$ als auch $\Sigma^* - L \in \mathbf{L}_0$ gilt. Die Menge aller rekursiven Sprachen über Σ wird mit $\mathbf{L}_{\text{rek}}(\Sigma)$, die Familie aller rekursiven Sprachen mit \mathbf{L}_{rek} bezeichnet.

Eine formale Sprache L ist also genau dann rekursiv, wenn sowohl die Sprache L selbst als auch ihr Komplement $\Sigma^* - L$ rekursiv aufzählbar sind. Damit bildet \mathbf{L}_{rek} aber auch die größte Sprachfamilie, für die das Problem $w \in L$ für alle $w \in \Sigma^*$ entscheidbar ist.

Das Problem $w \in L$ ist für rekursive Sprachen entscheidbar.

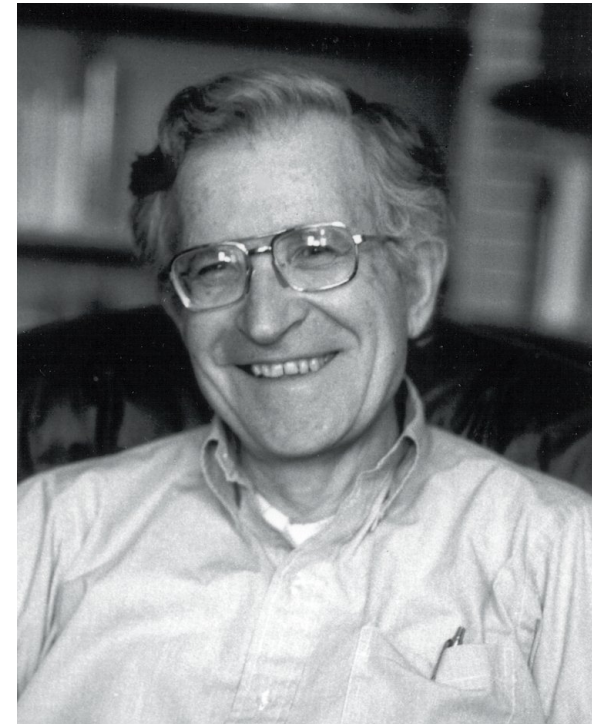
Entscheidbarkeit des Wortproblems für rekursive Sprachen

Das Problem $w \in L$ ist für rekursive Sprachen entscheidbar.

Betrachte Grammatik G mit $L(G) = L$ und Grammatik G' mit $L(G') = \Sigma^* - L$. Berechne für $n = 1, 2, \dots$ $SF(G, n)$ und $SF(G', n)$. Nach Definition von G und G' muss es ein n so geben, dass $w \in SF(G, n)$ (d.h., $w \in L$) oder $w \in SF(G', n)$ (d.h., $w \in \Sigma^* - L$).

Die Chomsky Hierarchie

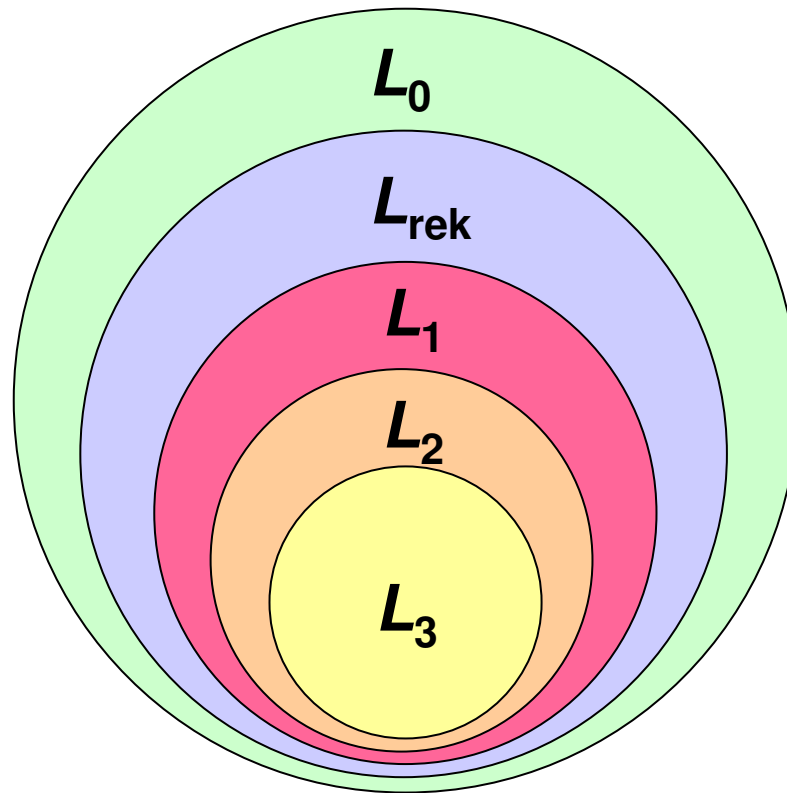
NOAM CHOMSKY (*1928)



1959 : On certain formal properties of grammars.
Information and Control 2 (1959), 137-167

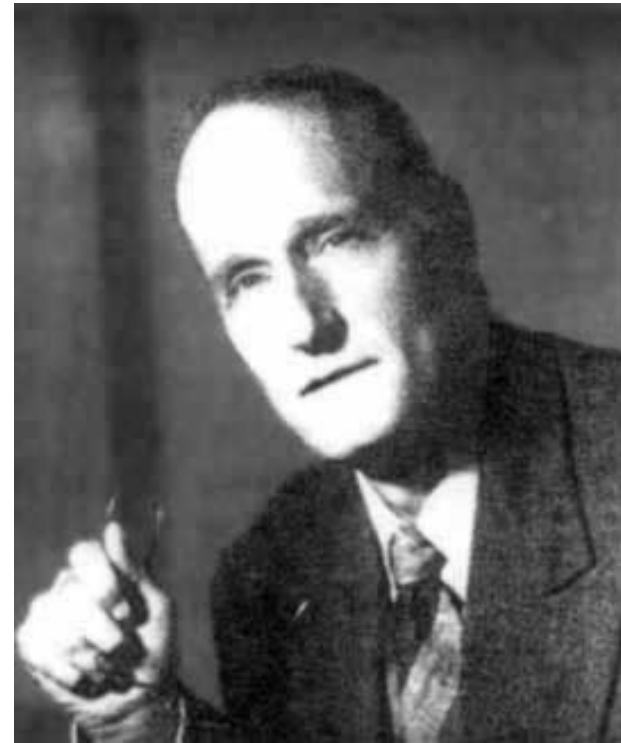
Die Chomsky Hierarchie

$$L_3 \subset L_2 \subset L_1 \subset L_{\text{rek}} \subset L_0$$



Endliche Automaten - Kleene

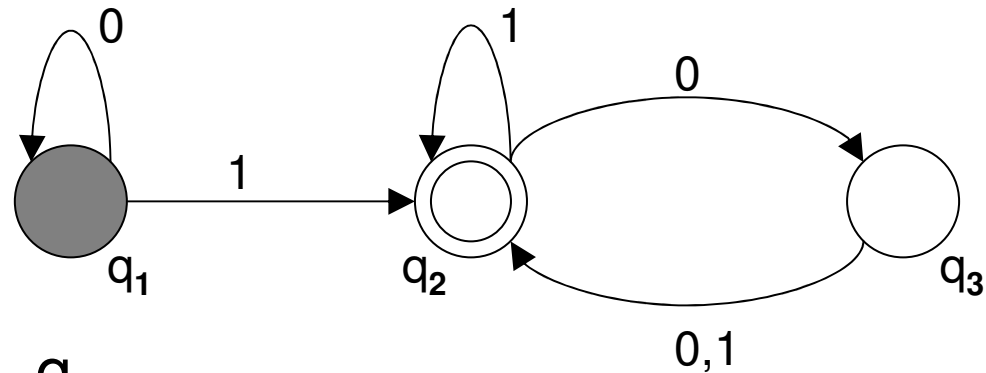
STEPHEN KLEENE (1909 - 1994)



1956: Representation of events in nerve nets and finite automata.
In: C.E. Shannon und J. McCarthy (eds.), Automata studies,
Princeton Univ. Press, 3-42

Endliche Automaten

Zustandsdiagramm:



Drei Zustände: q_1 , q_2 , q_3

Startzustand \bullet : q_1

Endzustand \odot : q_2

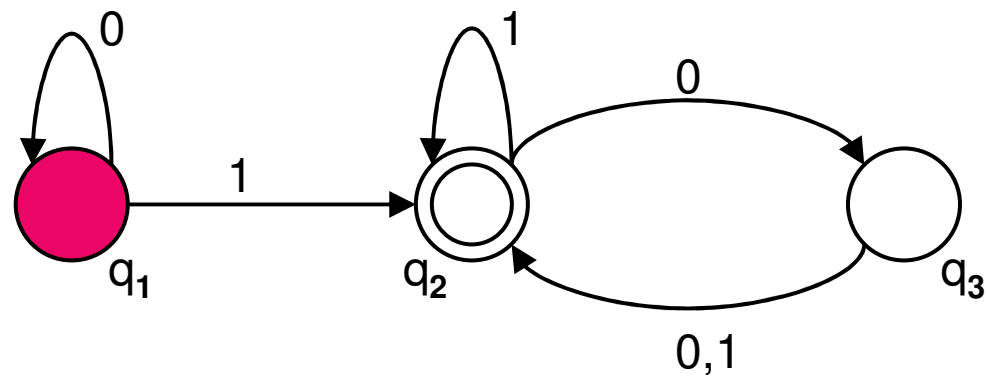
Transitionen (Übergänge): Pfeile

Eingabe: Wort

Ausgehend vom Startzustand liest der Automat M von links nach rechts Symbol für Symbol. Nach dem Lesen eines Symbols geht M in den nächsten Zustand über, indem er entlang der mit diesem Symbol markierten Kante geht. Nach dem Lesen des letzten Symbols wird der „Output“ erzeugt: Befindet sich M in einem Endzustand, wird das Wort akzeptiert; ansonsten wird das Wort nicht akzeptiert.

Endliche Automaten: Beispiel (1)

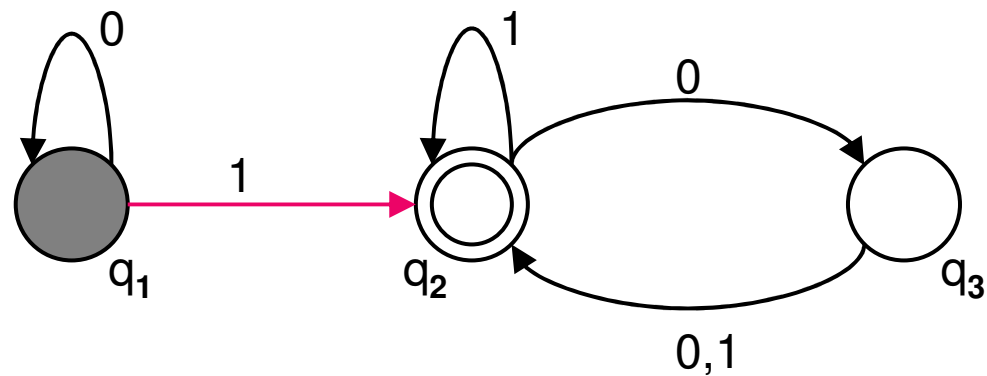
Eingabewort: 1101



Starte in Zustand q_1 .

Endliche Automaten: Beispiel (2)

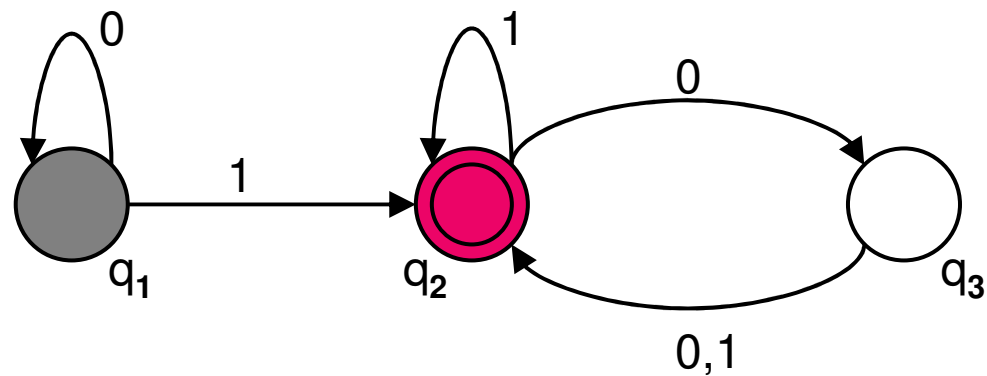
Wort: **1**101



Lies 1 und folge der mit 1 markierten Kante.

Endliche Automaten: Beispiel (3)

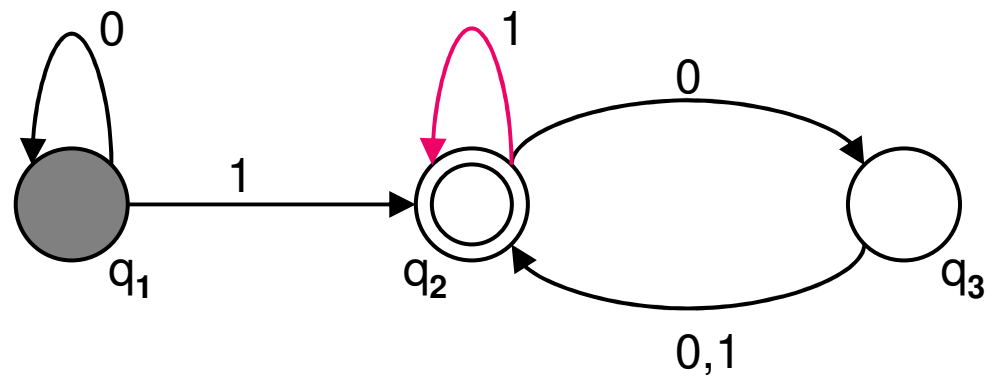
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2 .

Endliche Automaten: Beispiel (4)

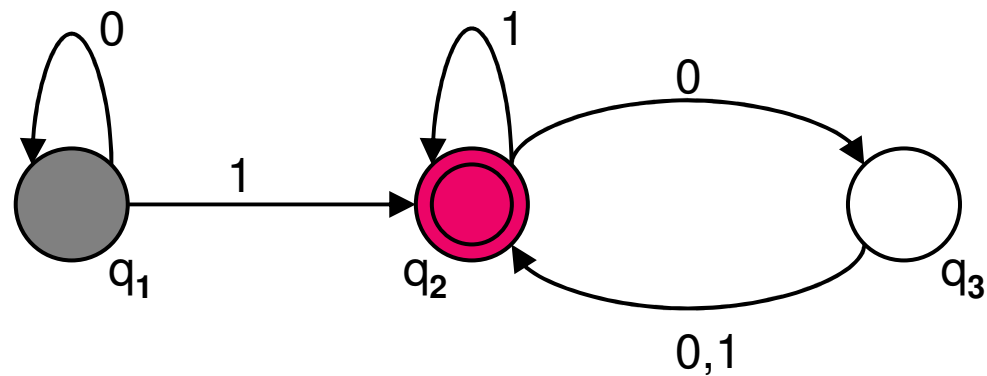
Wort: 1**1**01



Lies 1 und folge der mit 1 markierten Kante.

Endliche Automaten: Beispiel (5)

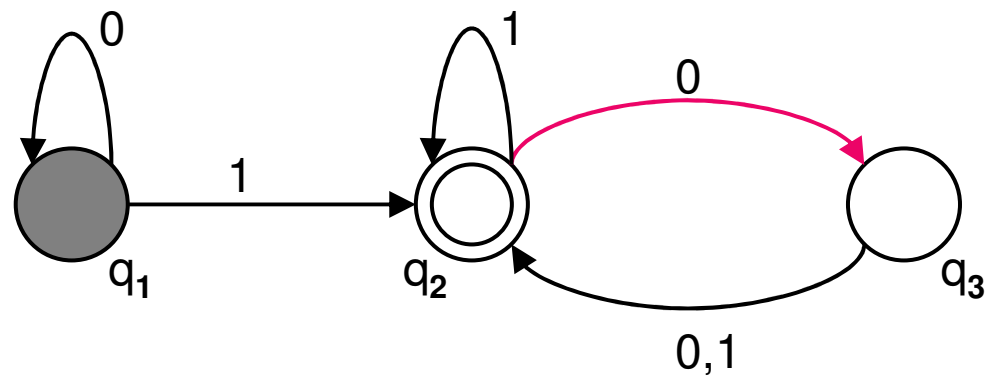
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2 .

Endliche Automaten: Beispiel (6)

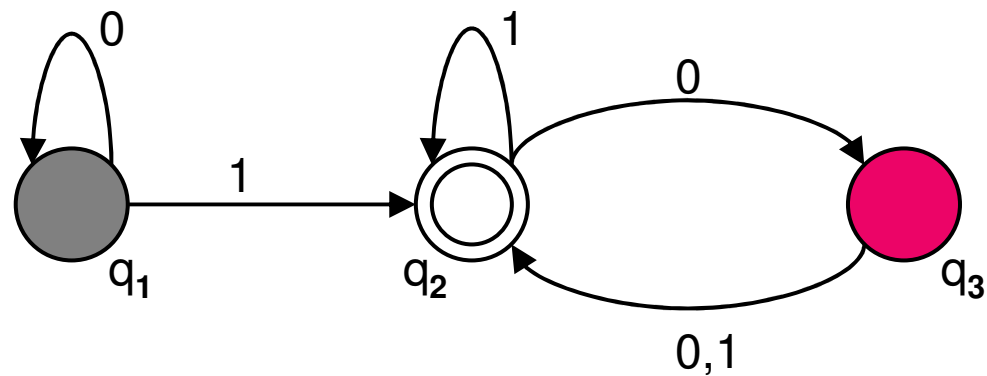
Wort: 1101



Lies 0 und folge der mit 0 markierten Kante.

Endliche Automaten: Beispiel (7)

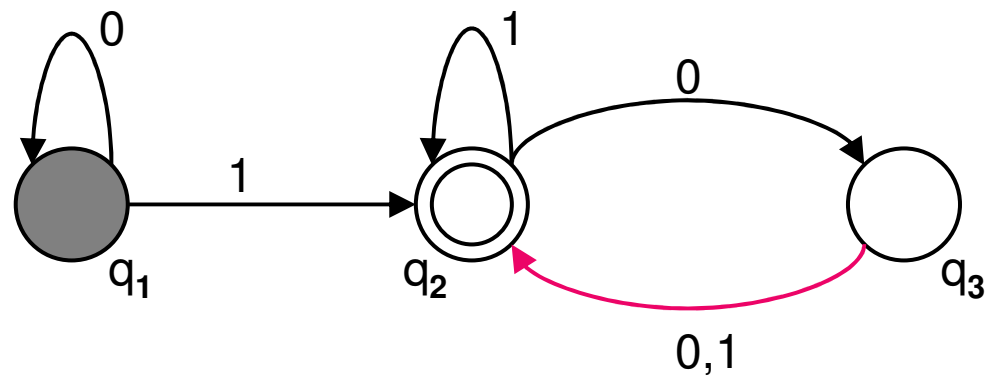
Wort: 1101



Lies 0 und folge der mit 0 markierten Kante zum Zustand q_3 .

Endliche Automaten: Beispiel (8)

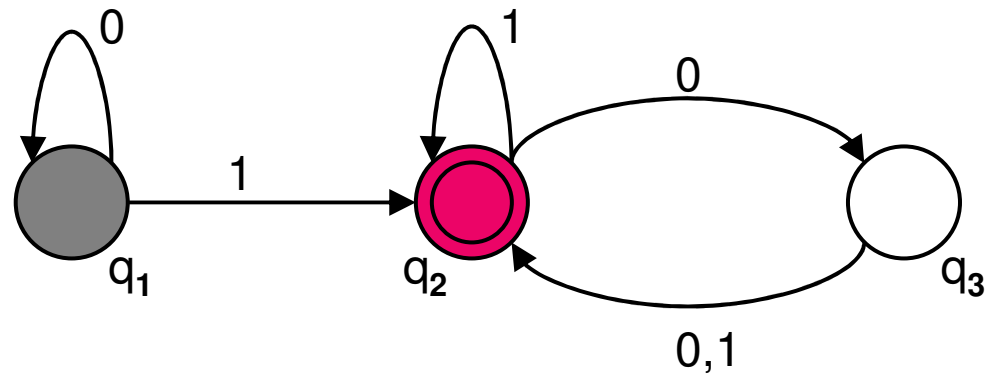
Wort: 110¹



Lies 1 und folge der mit 1 markierten Kante.

Endliche Automaten: Beispiel (9)

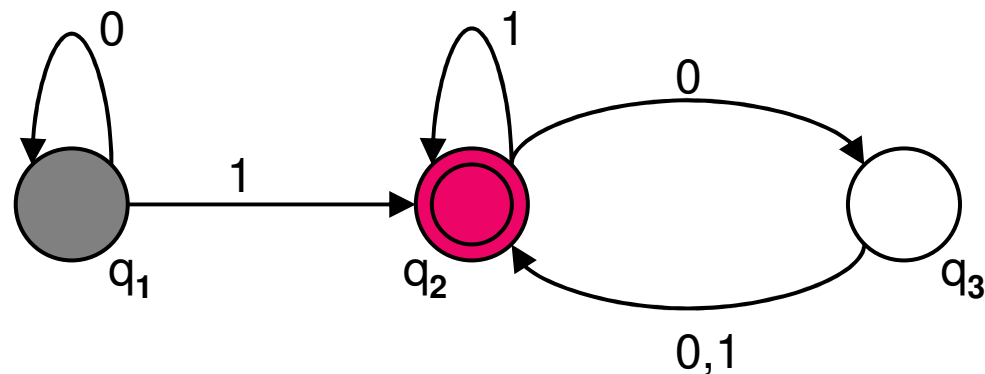
Wort: 110¹



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2 .

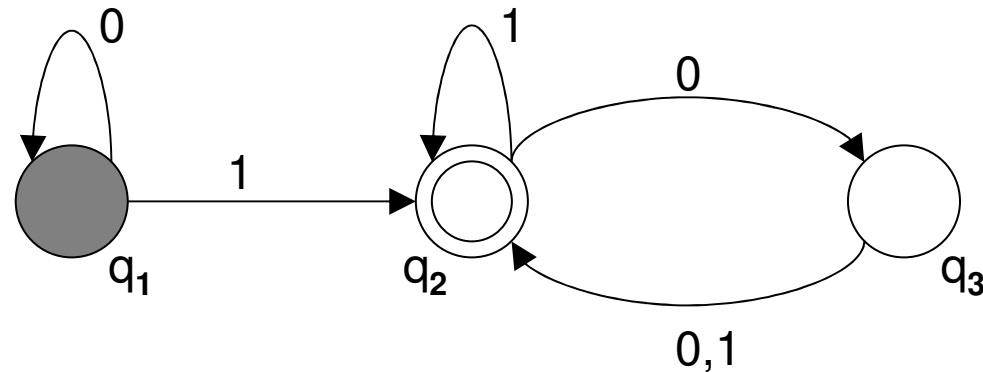
Endliche Automaten: Beispiel (10)

Wort: 1101



Das Wort 1101 wird akzeptiert, da sich der Automat am Ende des Eingabewortes in einem Endzustand befindet.

Endliche Automaten: Beispiel (11)



Akzeptierte Wörter:

1, 01, 11, 01010101, ...

Wörter, die mit 1 enden

Aber auch:

100, 0100, 110000, ...

Wörter, die mit einer geraden Anzahl von 0en nach der letzten 1 enden.

Nicht akzeptierte Wörter:

0, 10, 101000, ...

Akzeptierte Sprache:

$\{0\}^*\{1\}(\{1\}^*\{00,01\})^*\{1\}^*$

Endliche Automaten: formale Definition

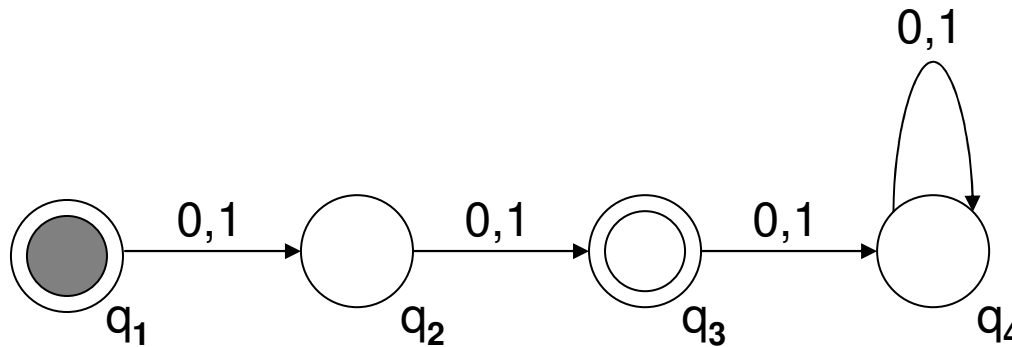
Ein **deterministischer endlicher Automat** (DEA) ist ein 5-Tupel

$$(Q, \Sigma, \delta, q_0, F)$$

wobei

- Q eine endliche Menge von **Zuständen**,
- Σ das **Alphabet**,
- $\delta: Q \times \Sigma \rightarrow Q$ die **Transitionsfunktion**,
- q_0 der **Startzustand** und
- $F \subseteq Q$ eine Menge von **Endzuständen** ist.

Endliche Automaten: Falle



Akzeptierte Sprache:
 $\{\epsilon, 00, 01, 10, 11\}$

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA und $q \in Q - F$ mit $\delta(q, a) = q$ für alle $a \in \Sigma$; dann heißt q **Falle**.

Um die Übersichtlichkeit zu erhöhen, können Fallen bei der Beschreibung endlicher Automaten weggelassen werden (allerdings nur, wenn ausdrücklich erlaubt).

Endliche Automaten: formale Definition (2)

Um das Verhalten eines DEA auf einer Zeichenkette formal zu beschreiben, erweitern wir die Übergangsfunktion δ auf beliebige Wörter aus Σ^* :

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Dann definieren wir die **erweiterte Übergangsfunktion** $\delta^*: Q \times \Sigma^* \rightarrow Q$ folgendermaßen:

$$\delta^*(q, \varepsilon) = q,$$

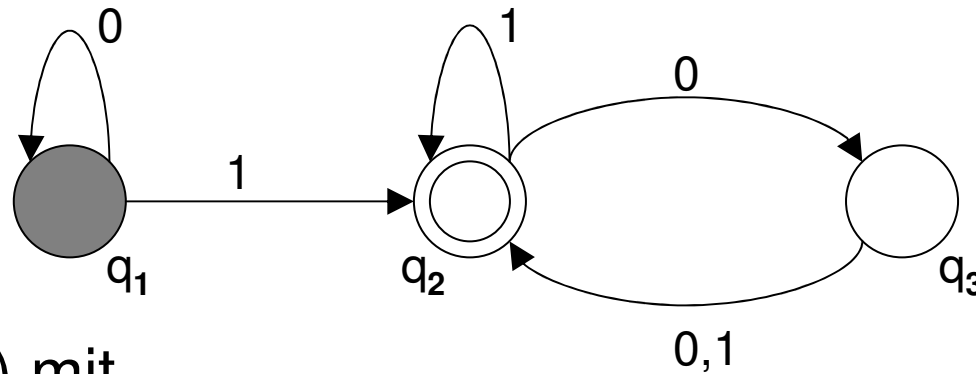
$$\delta^*(q, aw) = \delta^*(\delta(q, a), w) \quad \text{für alle } q \in Q, w \in \Sigma^*, a \in \Sigma.$$

Eine Zeichenkette $w \in \Sigma^*$ heißt vom DEA $M = (Q, \Sigma, \delta, q_0, F)$ akzeptiert, falls $\delta^*(q_0, w) = p$ für einen Zustand $p \in F$ gilt.

Die **von M akzeptierte Sprache**, bezeichnet mit $L(M)$, ist die Menge $\{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}$.

Endliche Automaten: Beispiel (12)

Beispiel:



$M = (Q, \Sigma, \delta, q_1, F)$ mit

$Q = \{ q_1, q_2, q_3 \},$

$\Sigma = \{ 0, 1 \},$

δ (gegeben durch die Übergangsmatrix),

q_1 Startzustand,

$F = \{ q_2 \}.$

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

$$\begin{aligned} \delta^*(q_1, 1101) &= \delta^*(\delta(q_1, 1), 101) = \delta^*(q_2, 101) = \delta^*(\delta(q_2, 1), 01) = \\ &\delta^*(q_2, 01) = \delta^*(\delta(q_2, 0), 1) = \delta^*(q_3, 1) = \delta^*(q_2, \epsilon) = q_2 \end{aligned}$$

$$L(M) = \{0\}^* \{1\} (\{1\}^* \{00, 01\})^* \{1\}^*$$

Minimalautomat

Zu jeder regulären Sprache L kann man effektiv einen DEA M mit einer minimalen Anzahl von Zuständen konstruieren, der bis auf die Umbenennung der Zustände eindeutig ist.

Beweis: s. einschlägige Literatur.

Nondeterminismus

MICHAEL O. RABIN (*1931)



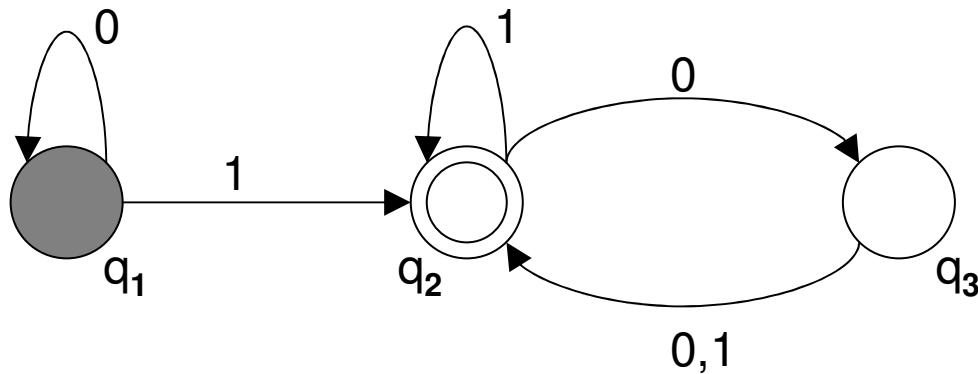
DANA SCOTT (*1932)



1959: *Finite Automata and Their Decision Problem*
IBM J. Research and Development, 3:114-125

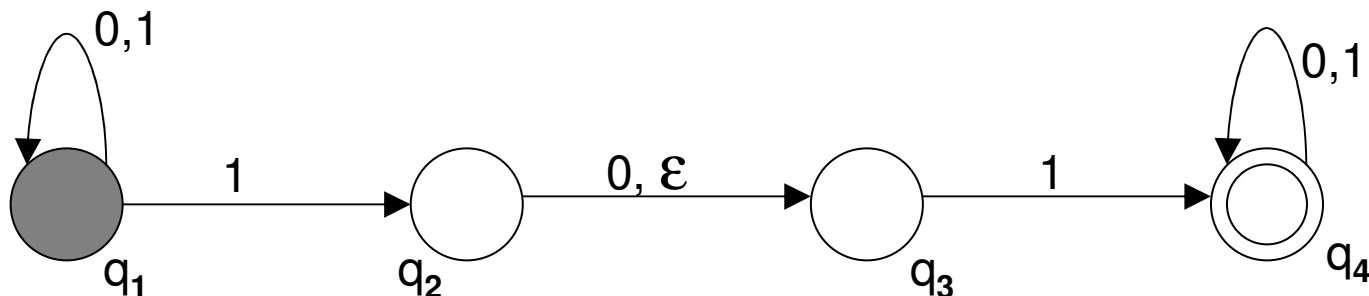
1976: Turing-Preis für Informatik

Nondeterminismus



DEA

Von einem Zustand aus gibt es mit ein und demselben Eingabesymbol genau einen Folgezustand.



NEA

Übergänge sind auch mit ε möglich (ε -Übergänge).

Von einem Zustand aus kann es mit ein und demselben Eingabesymbol mehrere Folgezustände geben.

NEA

Ein nichtdeterministischer endlicher Automat (NEA)
ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$

wobei

- Q eine endliche Menge von Zuständen,
- Σ das Alphabet,
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$ die Transitionsfunktion
- q_0 der Startzustand und
- $F \subseteq Q$ eine Menge von Endzuständen ist.

Äquivalenz von NEA und DEA - Beweis

Zu jedem nicht-deterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

NEA: $M = (Q, \Sigma, \delta, q_0, F)$

DEA: $M = (Q', \Sigma, \delta', q'_0, F')$ wobei

$$Q' = 2^Q$$

$$\delta'(q', a) = \bigcup_{q \in q'} \delta^*(q, a) \quad \text{für alle } q' \in Q', a \in \Sigma$$

$$q'_0 = \{q_0\}$$

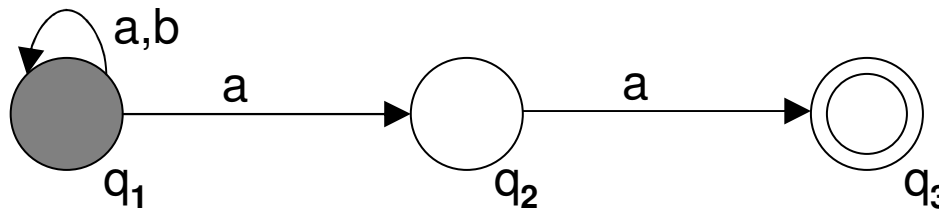
$$F' = \begin{cases} \{q' \in Q' \mid q' \cap F \neq \emptyset\} \cup \{q'_0\} & \text{falls } \varepsilon \in L(A) \\ \{q' \in Q' \mid q' \cap F \neq \emptyset\} & \text{sonst} \end{cases}$$

□

vom NEA zum DEA: Beispiel - Konstruktion

(für NEAs ohne ε -Übergänge)

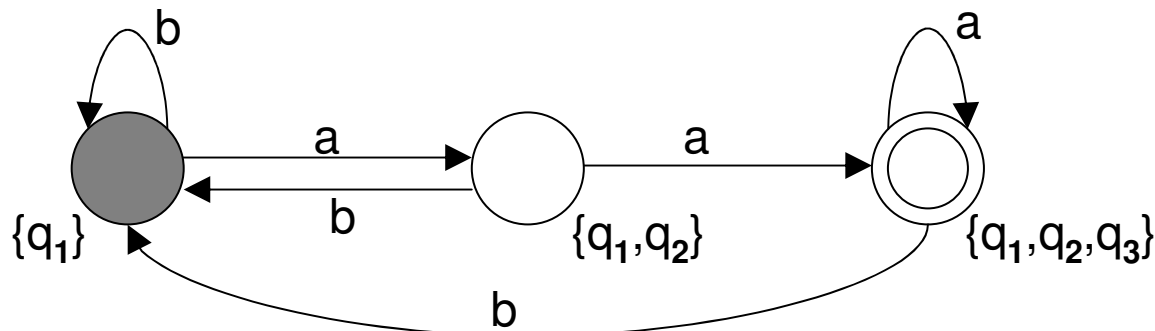
$$L(M) = \{ waa \mid w \in \{a,b\}^* \}$$



	δ'	a	b
SZ	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_1\}$
	$\{q_1, q_2\}$	$\{q_1, q_2, q_3\}$	$\{q_1\}$
EZ	$\{q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_1\}$

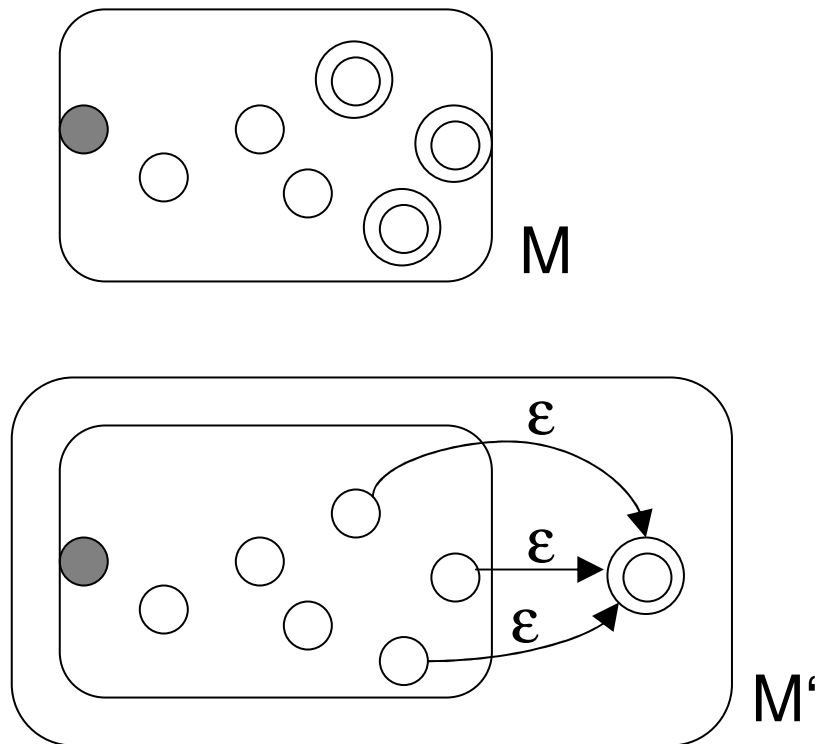
WICHTIG:

Bei Beispielen nur die notwendigen Zustände einführen, nicht alle!!



NEA mit einem Endzustand

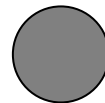
Zu jedem NEA M gibt es einen äquivalenten NEA M' mit $\text{card}(F) = 1$.



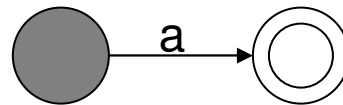
EA und reguläre Sprachen (1)

Zu jeder regulären Sprache R gibt einen endlichen Automaten M sodass $R = L(M)$

$L = \{ \}$

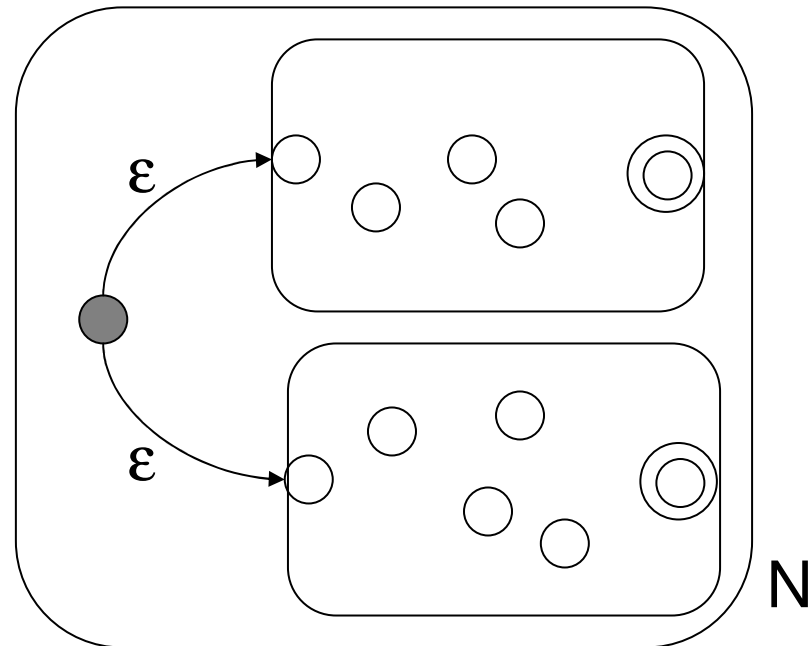
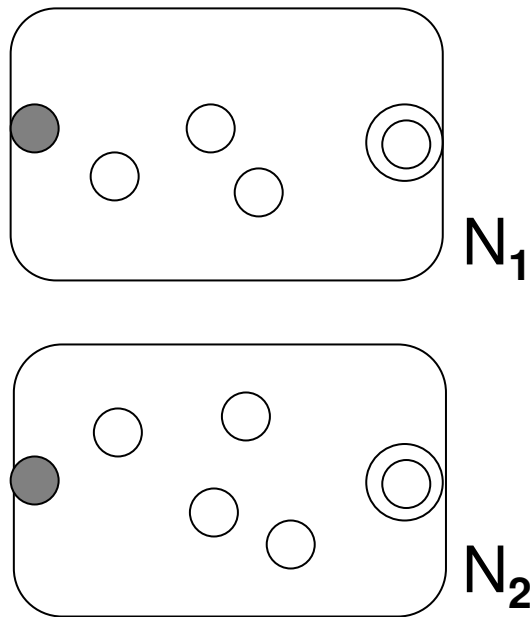


$L = \{a\}$ für $a \in \Sigma$



EA und reguläre Sprachen (2)

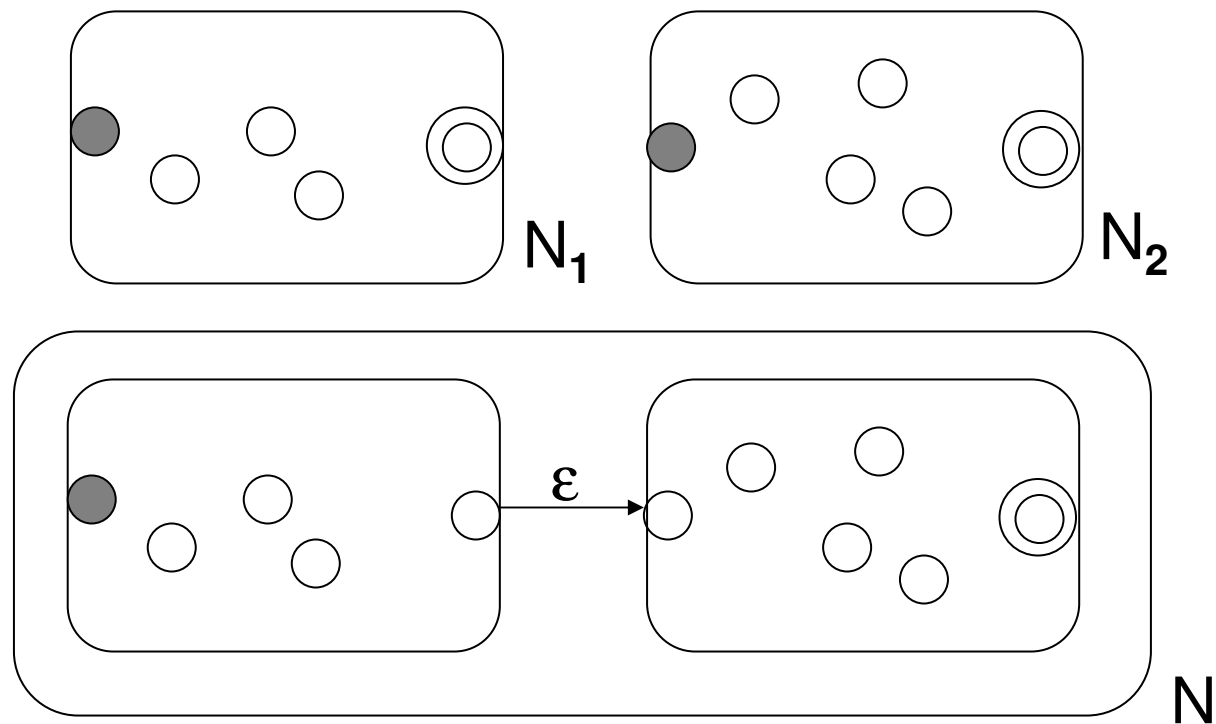
$$L = L_1 \cup L_2$$



EA und reguläre Sprachen (3)

$$L = L_1 \cdot L_2$$

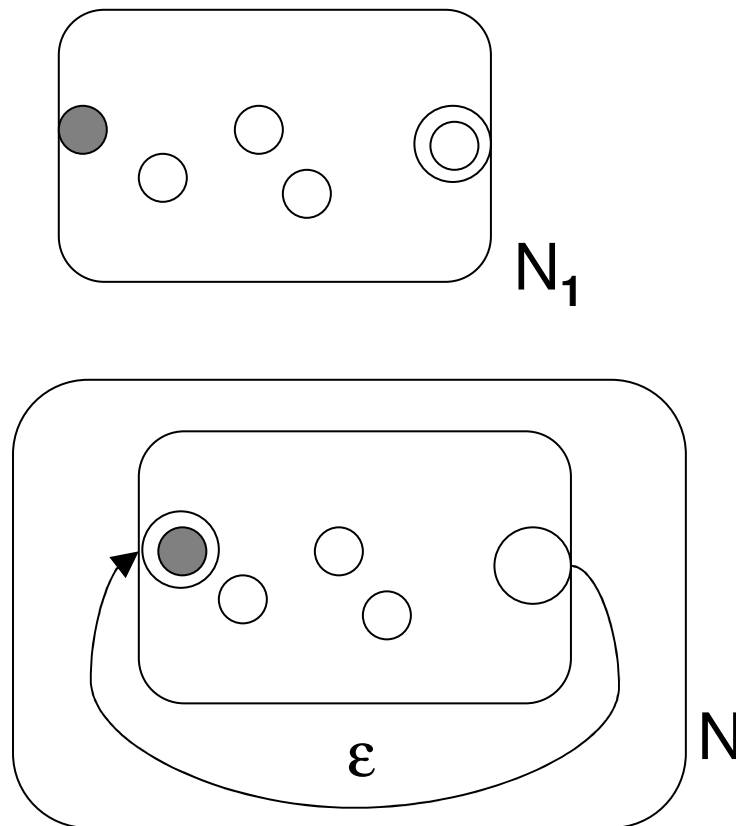
(O.B.d.A. besitzt N_1 nur einen Endzustand !)



EA und reguläre Sprachen (4)

$$L = (L_1)^*$$

(O.B.d.A. besitzt N_1 nur einen Endzustand !)

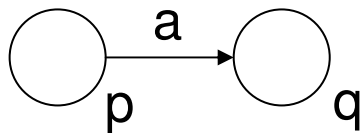


DEA und reguläre Grammatik

Zu jedem DEA M gibt es eine reguläre Grammatik G sodass $L(M) = L(G)$ und umgekehrt.

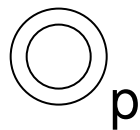
$$M = (Q, \Sigma, \delta, q_0, F)$$

$$G = (Q, \Sigma, P, q_0)$$



$$p \rightarrow aq$$

$$p, q \in Q \text{ und } a \in \Sigma$$
$$\delta(p, a) = q$$



$$p \rightarrow \varepsilon$$



Von DEA zu REG

Wird eine Sprache von einem DEA akzeptiert, dann ist sie regulär.

$$M = (\{ q_i \mid 1 \leq i \leq n \}, \Sigma, \delta, q_0, F)$$

R^k_{ij} Menge aller Wörter, mit denen man von q_i nach q_j gelangt, ohne einen Zwischenzustand mit Index größer als k zu durchlaufen.

$$R^0_{ij} = \begin{cases} \{a \in \Sigma \mid \delta(q_i, a) = q_j\} & \text{für } i \neq j \\ \{a \in \Sigma \mid \delta(q_i, a) = q_j\} \cup \{\varepsilon\} & \text{für } i = j \end{cases}$$

$$R^k_{ij} = R^{k-1}_{ij} \cup R^{k-1}_{ik} \cdot (R^{k-1}_{kk})^* \cdot R^{k-1}_{kj} \quad \text{für } k > 0$$

$$L(M) = \bigcup_{q_j \in F} R^n_{1j}$$

□

Reguläre Sprachen: Zusammenfassung

Beschreibungsmethoden für reguläre Sprachen:

Reguläre Mengen

Reguläre Grammatiken

DEA

NEA