# Formale Sprachen und Automaten

*Robin Rausch, Florian Maslowski*

*2. November 2022*

## Inhaltsverzeichnis

# 1 Grundlagen

## 1.1 Alphabet

Ein Alphabet $\Sigma$ ist eine nicht-leere Menge von Symbolen(Zeichen, Buchstaben). Beispiel: $\Sigma_{ab} = a, b$

## 1.2 Wort

Ein Wort $w$ über dem Alphabet $\Sigma$(Sigma) ist eine endliche Folge von Symbolen aus $\Sigma$. Das Wort $w = abaabab$ wurde beispielsweise aus dem Alphabet $\Sigma_{ab}$ gebildet.
Die Länge eines Wortes kann durch Betragsstriche angegeben werden. Beispiel: $|w| = 7$
Ebenso kann man die Anzahl bestimmter Symbole in einem Wort bestimmen: $|w|_b = 3$
Ein einzelnes Zeichen kann durch eckige Klammern angegeben werden: $w[2] = b$
Wörter können bliebig konkateniert werden(hintereinanderschreiben ohne abstand): $w_1 w_2 = abbabaab$ mit $w_1 = abba$ und $w_2 = baab$.
Wörter dürfen auch potenziert werden: $w^3 = abaababababaababababaabab = www$
Das leere Wort lautet $\varepsilon$.

## 1.3 Formale Sprachen

Eine formale Sprache $L$ über einem Alphabet $\Sigma$ ist eine Menge von Wörtern aus $\Sigma^* : L \subseteq \Sigma^*$. Eine Sprache kann sowohl endlich als auch unendlich sein.
Beispiel: $L_1 = \{w \in \Sigma^*_{bin} | \ |w| \geqslant 2 \wedge w[|w| - 1] = 1\}$ ist die Menge aller Binärwörter, an deren vorletzter Stelle 1 steht.
Das Produkt zweier formaler Sprachen: $L_1 \cdot L_2 = \{abac, abcb, bcac, bccb\}$ mit $L_1 = \{ab, bc\}$ und $L_2 = \{ac, cb\}$.
Sprachen können ebenfalls potenziert werden: $L^2 = \{ab, ba\} \cdot \{ab, ba\} = \{abab, abba, baab, baba\}$

## 1.4 Kleene Stern

Für ein Alphabet $\Sigma$ und eine formale Sprache $L \subseteq \Sigma^*$ ist der Operator Kleene Stern wie folgt definiert: $L^* = \bigcup_{n \in \mathbb{N}} L^n$.

Beispiel: Sei $L_1 = \{ab, ba\}$, dann $L^* = \{\varepsilon, ab, ba, abab, abba, baab, baba, ababab, ...\}$.

# 2 Reguläre Sprachen und endliche Ausdrücke

## 2.1 Reguläre Ausdrücke

Ein regulärer Ausdruck über $\Sigma$ beschreibt eine formale Sprache.
Die Menge aller regulären Ausdrücke über $\Sigma$ ist eine formale Sprache.

Beispiel: Sprache aller Wörter über $\Sigma_{abc}$, die nur aus genau zwei Symbolen bestehen:
Ausdruck: $r_1 = (a + b + c)(a + b + c)$
Sprache: $\mathcal{L}(r_1) = \{w \in \Sigma^*_{abc} \mid |w| = 2\}$

Operatoren:

| | | | |
|---|---|---|---|
| $r_1 + r_2$ | $\equiv$ | $r_2 + r_1$ | Kommutativität von $+$ |
| $(r_1 + r_2) + r_3$ | $\equiv$ | $r_1 + (r_2 + r_3)$ | Assoziativität von $+$ |
| $(r_1 r_2) r_3$ | $\equiv$ | $r_1 (r_2 r_3)$ | Assoziativität von $\cdot$ |
| $\emptyset r \equiv r \emptyset$ | $\equiv$ | $\emptyset$ | Absorbierendes Element für $\cdot$ |
| $\varepsilon r \equiv r \varepsilon$ | $\equiv$ | $r$ | Neutrales Element für $\cdot$ |
| $\emptyset + r$ | $\equiv$ | $r$ | Neutrales Element für $+$ |
| $(r_1 + r_2) r_3$ | $\equiv$ | $r_1 r_3 + r_2 r_3$ | Distributivität links |
| $r_1 (r_2 + r_3)$ | $\equiv$ | $r_1 r_2 + r_1 r_3$ | Distributivität rechts |
| $r + r$ | $\equiv$ | $r$ | Idempotenz von $+$ |
| $(r^*)^*$ | $\equiv$ | $r^*$ | Idempotenz von $^*$ |
| $\emptyset^*$ | $\equiv$ | $\varepsilon$ | |
| $\varepsilon^*$ | $\equiv$ | $\varepsilon$ | |
| $\varepsilon + r^* r$ | $\equiv$ | $r^*$ | |
| $(\varepsilon + r)^*$ | $\equiv$ | $r^*$ | |
| $r^* r$ | $\equiv$ | $r r^*$ | |

Nicht alle Operatoren sind für alle Typen zulässig:

| | Vereinigung | Konkatenation | Potenz | Kleene-Stern |
|---|---|---|---|---|
| Wörter | ✗ | $w_1 \cdot w_2$ | $w^n$ | ✗ |
| Sprachen | $L_1 \cup L_2$ | $L_1 \cdot L_2$ | $L^n$ | $L^*$ |
| Reguläre Ausdrücke | $r_1 + r_2$ | $r_1 \cdot r_2$ | ✗ | $r^*$ |

## 2.2 Endliche Automaten

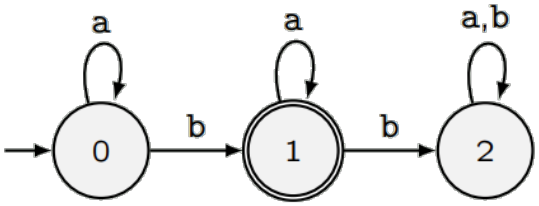
Endliche Automaten sind eine andere Darstellung einer regulären Sprache. Endliche Ausdrücke lassen sich in Reguläre Ausdrücke umformen. Genauso auch anders herum.
Endliche Automaten erkennen regulären Sprachen. Endliche Ausdrücke lassen sich in Reguläre Ausdrücke umformen. Genauso auch anders herum.
Endliche Automaten lassen sich sowohl deterministisch als auch nicht-deterministisch darstellen.

### 2.2.1 Deterministische endliche Automaten(DEA)

Ein DEA hat endlich viele Zustände. Jeder mögliche Übergang muss hierbei behandelt werden können. D.h. für das Alphabet $\Sigma_{ab}$ muss von jedem Zustand sowohl ein $a$, als auch ein $b$ Übergang gegeben sein. Der Automat beginnt im Startzustand und muss im Endzustand enden. Wenn der Automat sich in einem Nicht-Endzustand befindet, befindet sich das Wort nicht in der Sprache, welche vom Automaten abgebildet wird.
Ein DEA hat endlich viele Zustände. Jeder mögliche Übergang muss hierbei behandelt werden können. D.h. für das Alphabet $\Sigma_{ab}$ muss von jedem Zustand sowohl ein $a$, als auch ein $b$

Übergang gegeben sein. Er terminiert wenn das Wort zu ende und Endzustand erreicht ist.
Der DEA lässt sich durch folgendes 5-Tupel darstellen:
$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ mit den Komponenten:
$Q$ ist eine endliche Menge von Zuständen
$\Sigma$ ist ein endliches Alphabet
$\delta : Q \times \Sigma \to Q$ ist die Übergangsfunktion
$q_0 \in Q$ ist der Startzustand
$F \subseteq Q$ ist die Menge der Endzustände

Beispiel:




$\mathcal{A}_{\mathtt{b}} = (Q, \Sigma, \delta, q_0, F)$ mit

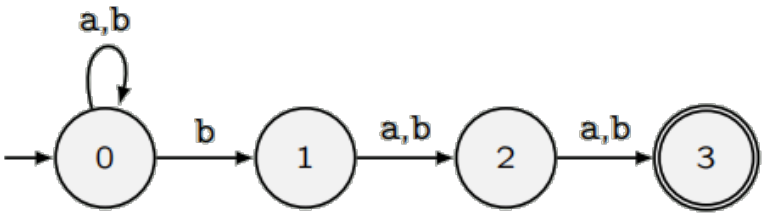
- $Q = \{0, 1, 2\}$
- $\Sigma = \Sigma_{\mathtt{ab}}$
- $\delta(0, \mathtt{a}) = 0; \delta(0, \mathtt{b}) = 1, \delta(1, \mathtt{a}) = 1; \delta(1, \mathtt{b}) = \delta(2, \mathtt{a}) = \delta(2, \mathtt{b}) = 2$
- $q_0 = 0$
- $F = \{1\}$

Zustand 2: „Mülleimerzustand“ (junk state), d.h. kein Wort wird mehr akzeptiert

### 2.2.2 Nicht-deterministische endliche Automaten(NEA)

Ein NEA hat endlich viele Zustände. Nicht jeder mögliche Übergang muss hierbei behandelt werden. D.h. für das Alphabet $\Sigma_{ab}$ reicht es, nur den $a$-Übergang, bzw. nur den $b$-Übergang zu besitzen. Der Automat beginnt im Startzustand und muss im Endzustand enden. Wenn der Automat sich in einem Nicht-Endzustand befindet, befindet sich das Wort nicht in der Sprache, welche vom Automaten abgebildet wird. Zudem gibt es $\varepsilon$-Übergänge, diese können jederzeit verwendet werden ohne ein Eingabesymbol.
Der NEA lässt sich durch folgendes 5-Tupel darstellen:
$\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ mit den Komponenten:
$Q$ ist eine endliche Menge von Zuständen
$\Sigma$ ist ein endliches Alphabet
$\Delta$ ist eine Relation über $Q \times (\Sigma \cup \{\varepsilon\}) \times Q$
$q_0 \in Q$ ist der Startzustand
$F \subseteq Q$ ist die Menge der Endzustände

Beispiel:




$\mathcal{A}_n = (Q, \Sigma, \Delta, q_0, F)$ mit

$Q = \{0, 1, 2, 3\}$

$\Sigma = \Sigma_{ab}$

$\Delta = \{(0, a, 0), (0, b, 0), (0, b, 1),$
$(1, a, 2), (1, b, 2),$
$(2, a, 3), (2, b, 3)\}$

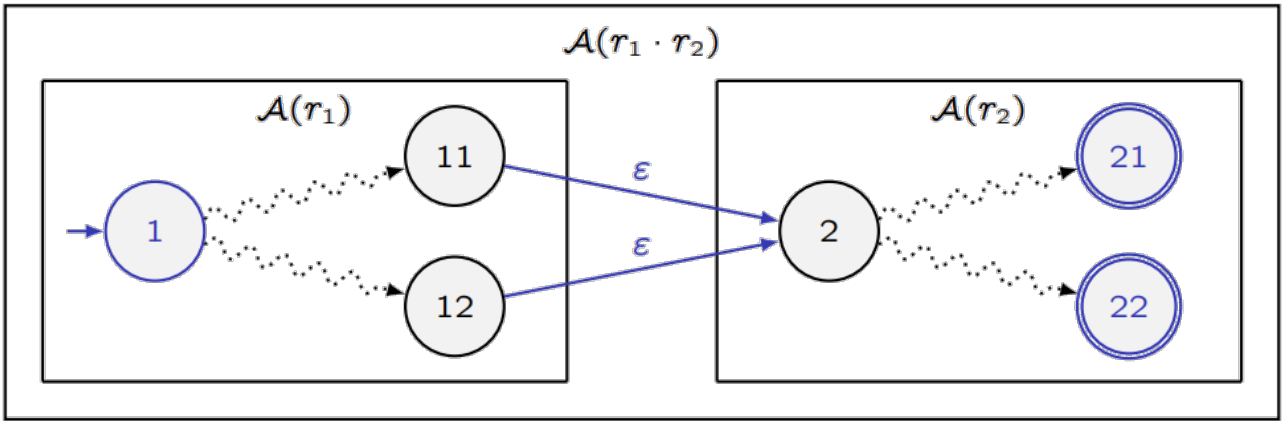
$q_0 = 0$

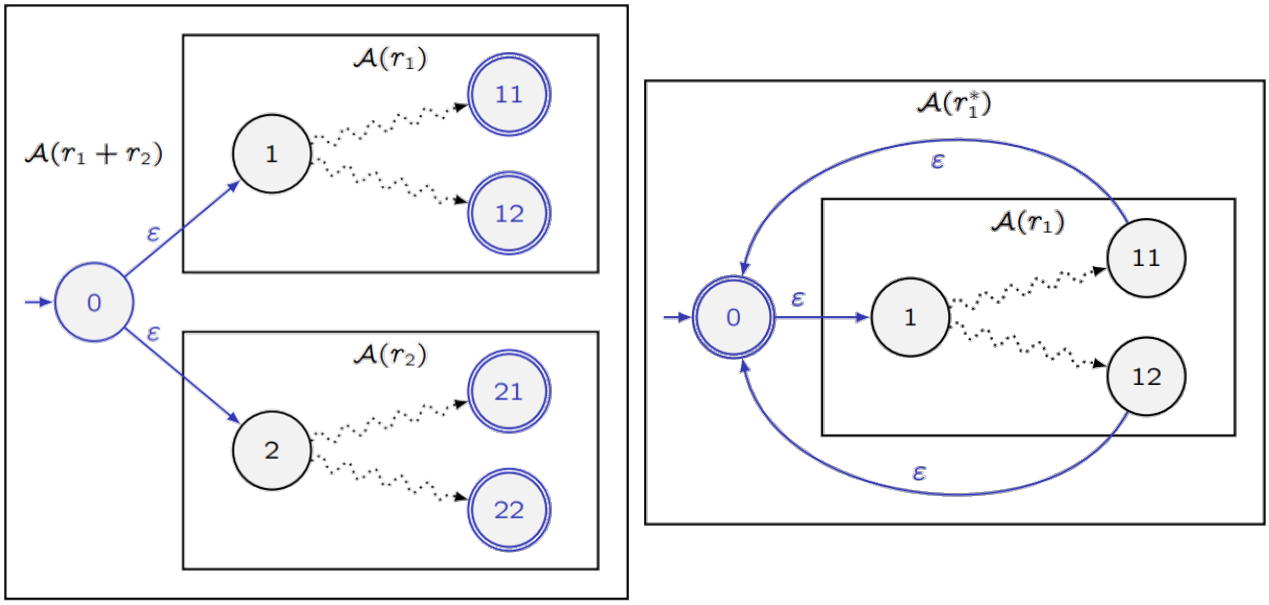
$F = \{3\}$

| $\mathcal{A}_n$ | | a | b | $\varepsilon$ |
|---|---|---|---|---|
| $\rightarrow$ | 0 | {0} | {0, 1} | {} |
| | 1 | {2} | {2} | {} |
| | 2 | {3} | {3} | {} |
| * | 3 | {} | {} | {} |

### 2.2.3 Endliche Automaten und reguläre Ausdrücke

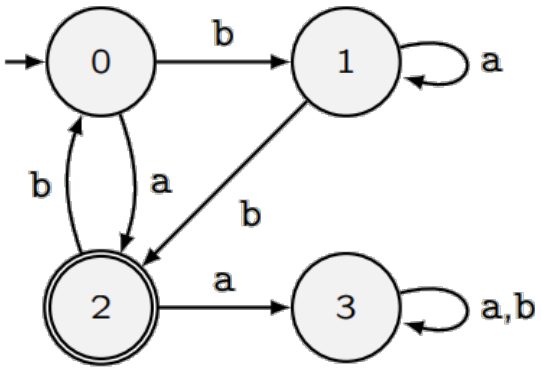
Man kann NEA und RA zusammenführen zu einem Automaten:






### 2.2.4 Transformation DEA zu RA

Um einen DEA zu einem RA umzuformen müssen zuerst die Gleichungen der jeweiligen Zustände aufgestellt und vereinfacht werden:




- $L_0 \equiv aL_2 + bL_1$
- $L_1 \equiv aL_1 + bL_2$
- $L_2 \equiv aL_3 + bL_0 + \varepsilon$
- $L_3 \equiv (a + b)L_3$

$$
\begin{aligned}
L_3 &\equiv (a+b)L_3 + \emptyset && \text{[neutrales Element]}\\
&\equiv (a+b)^*\emptyset && \text{[Arden]}\\
&\equiv \emptyset && \text{[absorbierendes Element]}\\
L_2 &\equiv a\emptyset + bL_0 + \varepsilon && \text{[einsetzen } L_3\text{]}\\
&\equiv \emptyset + bL_0 + \varepsilon && \text{[absorbierendes Element]}\\
&\equiv bL_0 + \varepsilon && \text{[neutrales Element]}\\
L_1 &\equiv aL_1 + b(bL_0 + \varepsilon) && \text{[einsetzen } L_2\text{]}\\
&\equiv a^*b(bL_0 + \varepsilon) && \text{[Arden]}
\end{aligned}
$$

Folglich kann man die gekürzten Gleichungen in einander einsetzen um bis zum Anfangszustand zu kommen:

- $L_0 \equiv aL_2 + bL_1$
- $L_1 \equiv a^*b(bL_0 + \varepsilon)$
- $L_2 \equiv bL_0 + \varepsilon$
- $L_3 \equiv \emptyset$

Da $L_2$ der Endzustand ist, bekommt die Gleichung $\varepsilon$ hinzuaddiert!
Der vereinfachte Anfangszustand $L_0$ ist dann der RA zum zugehörigen DEA:

$$\begin{aligned}
L_0 &\equiv a(bL_0 + \varepsilon) + b(a^*b(bL_0 + \varepsilon)) && \text{[einsetzen } L_1, L_2\text{]} \\
&\equiv abL_0 + a + ba^*bbL_0 + ba^*b && \text{[Distributivgesetz]} \\
&\equiv (ab + ba^*bb)L_0 + a + ba^*b && \text{[Kommutativ-,Distributivgesetz]} \\
&\equiv (ab + ba^*bb)^*(a + ba^*b) && \text{[Arden]}
\end{aligned}$$

Ergebnis: $\mathcal{L}(\mathcal{A}) = \mathcal{L}((ab + ba^*bb)^*(a + ba^*b))$

#### 2.2.5 Minimierung

### 2.3 Nicht-reguläre Sprachen und das Pumping-Lemma

### 2.4 Eigenschaften regulärer Sprachen

## 3 Chomsky Grammatiken und kontextfreie Sprachen

Gramatiken erzeugen formale Sprachen dar.
G=($N, \sum, P, S$) mit:

**N** Nichtterminalsymbole. Diese können für Regeln verwendet werden, aber dürfen nicht selbst im abgeleiteten Wort stehen.

**P** Ableitungsregeln. Bsp.: P=$\{S \to Aa|\varepsilon, A \to a\}$

**S** Startsymbol (ist nichtterminel)

### 3.1 Typ0 unbeschränkt

### 3.2 Typ1 Monoton

$\alpha \to \beta$ mit $|\alpha| \leq |\beta|$ und Ausnahme $S \to \varepsilon$, wenn S auf keiner rechten Seite ist.

### 3.3 Typ2 Kontextfreie

A $\rightarrow \beta$ mit A $\in$ N und $\beta \in V^*$

### 3.4 Typ3 rechtsregulär/-linear

A $\rightarrow$ cB mit A $\in$ N; B $\in$ N $\cup \{\varepsilon\}$; c $\in \sum \cup \{\varepsilon\}$

### 3.5 Chomskynormalform CNF

Ohne $\varepsilon$ und unnötige Regeln und Terminalsymbole.

## 4 Turing Maschine

Automaten mit endlosem Einleseband.
Terminiert wenn Endzustand erreicht und Einleseband nicht verschiebbar ist.
M=(Q, $\sum, \Gamma, \Delta, q_0$, F) mit:

$\Gamma$ Vereinigung aus mindestens Blank-Symbol und Terminalsymbole: $\Gamma \supseteq \sum \cup \{\square\}$

$\Delta$ Übergangsrelationen Syntax: IST-Zustand IST-Inhalt Neuer-Inhalt Verschiebung Neuer-Zustand Bsp.: $0 \begin{pmatrix} a \\ \square \end{pmatrix} \begin{pmatrix} a \\ a \end{pmatrix} \begin{pmatrix} r \\ l \end{pmatrix} 1$

**F** Endzustände

## 5 Entscheidbarkeit

## 6 Berechenbarkeit

## 7 Komplexität