

Formale Sprachen und Automaten

Robin Rausch, Florian Maslowski

6. November 2022

Inhaltsverzeichnis

1 Grundlagen	1
1.1 Alphabet	1
1.2 Wort	1
1.3 Formale Sprachen	1
1.4 Kleene Stern	1
2 Reguläre Sprachen und endliche Ausdrücke	1
2.1 Reguläre Ausdrücke	1
2.2 Endliche Automaten	2
2.2.1 Deterministische endliche Automaten(DEA)	2
2.2.2 Nicht-deterministische endliche Automaten(NEA)	3
2.2.3 Endliche Automaten und reguläre Ausdrücke	4
2.2.4 Transformation DEA zu RA	5
2.2.5 Minimierung	6
2.3 Nicht-reguläre Sprachen und das Pumping-Lemma	6
2.4 Eigenschaften regulärer Sprachen	6
3 Chomsky Grammatiken und kontextfreie Sprachen	6
3.1 Typ0 unbeschränkt	6
3.2 Typ1 Monoton	6
3.3 Typ2 Kontextfreie	7
3.4 Typ3 rechtsregulär/-linear	7
3.5 Cocke-Younger-Kasami(CYK)-Algorithmus	7
3.6 Chomskynormalform CNF	7
4 Turing Maschine	8
5 Entscheidbarkeit	8
5.1 Entscheidbar	8
5.2 Unentscheidbar	8
5.3 Semi-entscheidbar	8
6 Berechenbarkeit	8
7 Komplexität	8

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	\times	$w_1 \cdot w_2$	w^n	\times
Sprachen	$L_1 \cup L_2$	$L_1 \cdot L_2$	L^n	L^*
Reguläre Ausdrücke	$r_1 + r_2$	$r_1 \cdot r_2$	\times	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 Σ_{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 Σ_{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

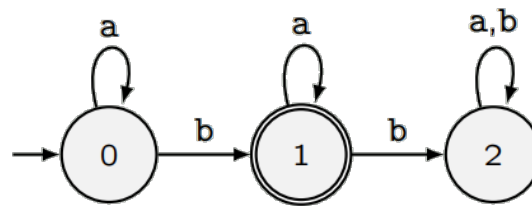
Σ ist ein endliches Alphabet

$\delta : Q \times \Sigma \rightarrow Q$ ist die Übergangsfunktion

$q_0 \in Q$ ist der Startzustand

$F \subseteq Q$ ist die Menge der Endzustände

Beispiel:



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

■ $Q = \{0, 1, 2\}$

■ $\Sigma = \Sigma_{ab}$

■ $\delta(0, a) = 0; \delta(0, b) = 1, \delta(1, a) = 1; \delta(1, b) = \delta(2, a) = \delta(2, b) = 2$

■ $q_0 = 0$

■ $F = \{1\}$

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 Σ_{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 ε -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

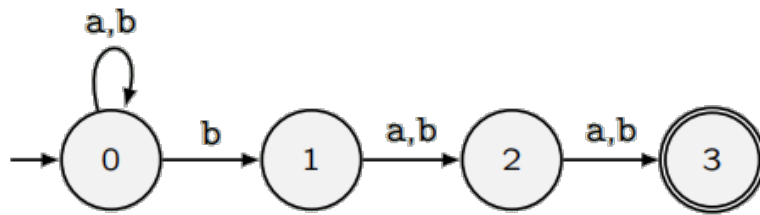
Σ ist ein endliches Alphabet

Δ 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

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

$\Sigma = \Sigma_{ab}$

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

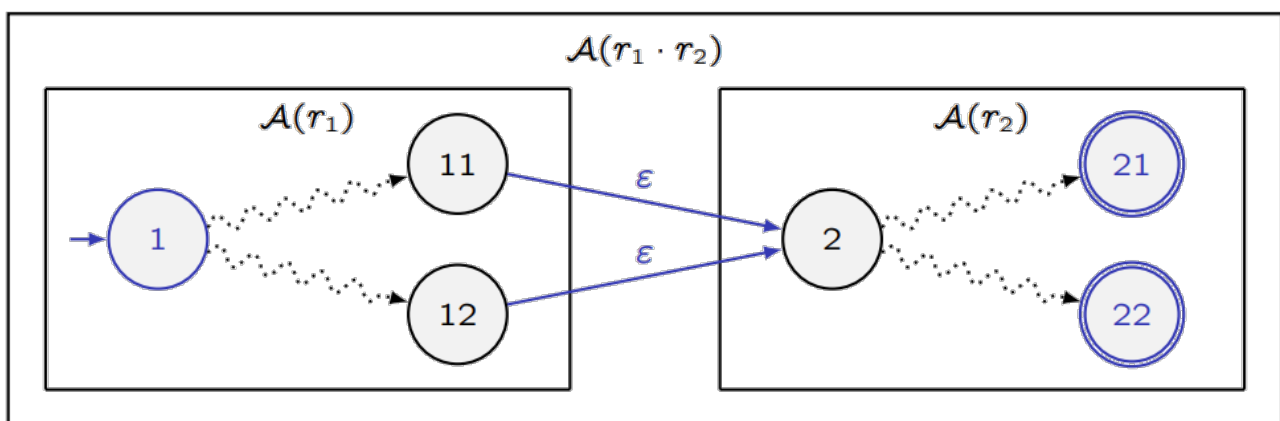
$q_0 = 0$

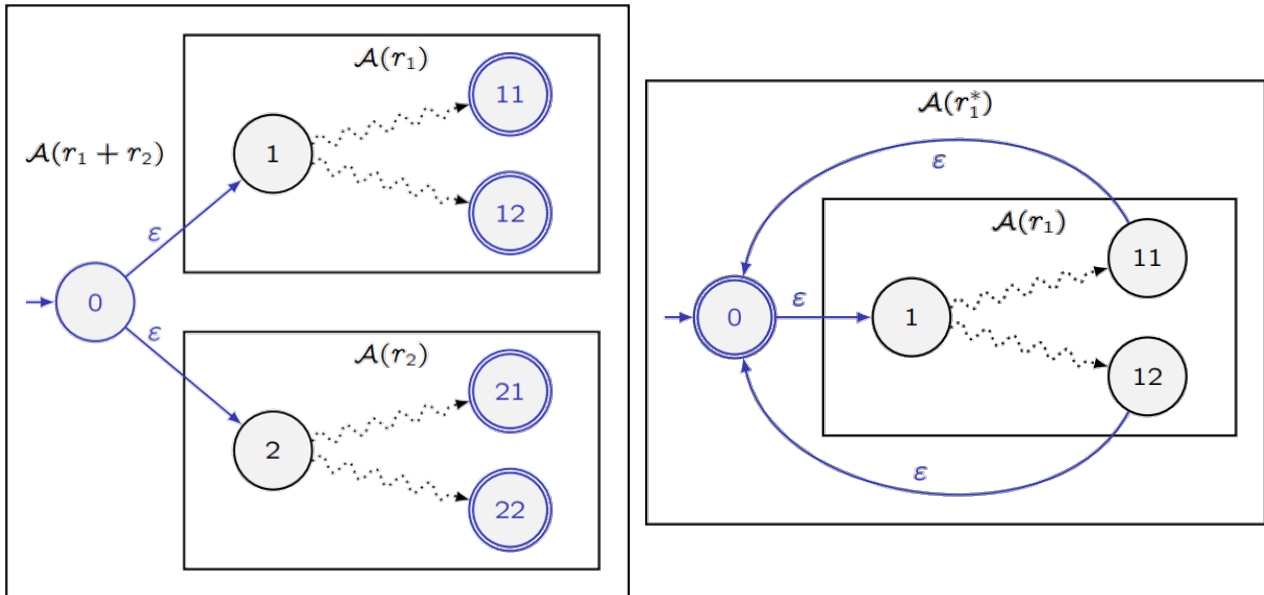
$F = \{3\}$

\mathcal{A}_n	a	b	ϵ
\rightarrow 0	{0}	{0, 1}	{}
1	{2}	{2}	{}
2	{3}	{3}	{}
* 3	{}	{}	{}

2.2.3 Endliche Automaten und reguläre Ausdrücke

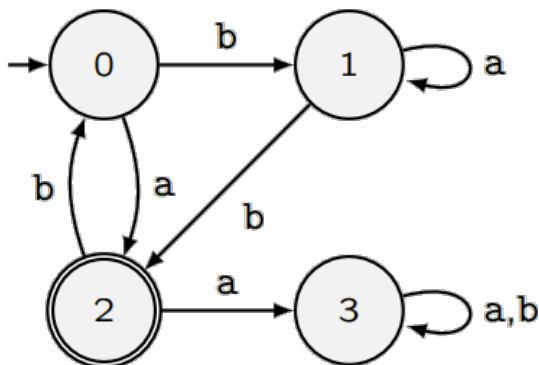
Man kann NEA und RA zusammenführen zu einem Automaten:





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:



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

$$\begin{aligned} L_3 &\equiv (a + b)L_3 + \emptyset \\ &\equiv (a + b)^*\emptyset \\ &\equiv \emptyset \end{aligned}$$

$$\begin{aligned} L_2 &\equiv a\emptyset + bL_0 + \varepsilon \\ &\equiv \emptyset + bL_0 + \varepsilon \\ &\equiv bL_0 + \varepsilon \end{aligned}$$

$$\begin{aligned} L_1 &\equiv aL_1 + b(bL_0 + \varepsilon) \\ &\equiv a^*b(bL_0 + \varepsilon) \end{aligned}$$

[neutrales Element]

[Arden]

[absorbierendes Element]

[einsetzen L_3]

[absorbierendes Element]

[neutrales Element]

[einsetzen L_2]

[Arden]

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 ε hinzuaddiert!

Der vereinfachte Anfangszustand L_0 ist dann der RA zum zugehörigen DEA:

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

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

2.2.5 Minimierung

2.3 Nicht-reguläre Sprachen und das Pumping-Lemma

2.4 Eigenschaften regulärer Sprachen

3 Chomsky Grammatiken und kontextfreie Sprachen

Grammatiken erzeugen formale Sprachen dar.

$G=(N, \Sigma, P, S)$ mit:

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

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

S Startsymbol (ist nichtterminel)

3.1 Typ0 unbeschränkt

3.2 Typ1 Monoton

$\alpha \rightarrow \beta$ mit $|\alpha| \leq |\beta|$ und Ausnahme $S \rightarrow \varepsilon$, 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 $A \in N$; $B \in N \cup \{\epsilon\}$; $c \in \sum \cup \{\epsilon\}$

3.5 Cocke-Younger-Kasami(CYK)-Algorythmus

Entscheidet Wortproblem für kontextfreie Grammatiken in CNF. Bsp.:

$w = abacba$

Regeln:

$S \rightarrow a$
 $B \rightarrow b$
 $B \rightarrow c$
 $S \rightarrow SA$
 $A \rightarrow BS$
 $B \rightarrow BB$
 $B \rightarrow BS$

$i \setminus j$	1	2	3	4	5	6
1	S	$\{\}$	S	$\{\}$	$\{\}$	S
2		B	A, B	B	B	A, B
3			S	$\{\}$	$\{\}$	S
4				B	B	A, B
5					B	A, B
6						S
$w =$	a	b	a	c	b	a

Umgekehrte Regeln:

$SA \leftarrow S$
 $BS \leftarrow A, B$
 $BB \leftarrow B$

1. Wort unter Tabelle schreiben
2. Umgekehrte Regeln bilden (optional)
3. Herleitungen in Tabellen-Diagonale eintragen
4. Herleitungen der Teilwörter als Menge eintragen (alle Möglichkeiten)

3.6 Chomskynormalform CNF

Zur Entscheidung des Wortproblems. Nur Regeln mit Syntax:

1. $A \rightarrow BC$ oder 2. $A \rightarrow a$ mit $A, B, C \in \text{Nichtterminalsymbole} \wedge a \in \text{Terminalsymbole}$ und $S \rightarrow \epsilon$, wenn S auf keiner Rechten Seite ist.

Vorgehen:

1. Epsilon-Regeln entfernen
 - (a) Erstelle Liste L mit $A \rightarrow \epsilon$ -Regeln und allen Regeln, die auf Nichtterminalsymbole in L zeigen.
 - (b) Ergänze rechte Seite der Regeln mit sich selbst mit eingesetztem ϵ für Nichtterminalsymbole $\in L$
 - (c) Wenn $S \in L$ füge Regel $S_0 \rightarrow S|\epsilon$ ein
2. Kettenregel entfernen
 - (a) Erstelle Listen für Kettenregeln ausgehend von jeweiligen Nichtterminalen

- (b) Ergänze Regeln mit allen Kettenregeln
- (c) Regeln kürzen
- 3. überflüssige Symbole entfernen
- 4. Einzelne Nichtterminalsymbole auf rechter Seite ersetzen
- 5. Rechte Seite mit Hilfssymbolen kürzen

4 Turing Maschine

Automaten mit endlosem Einleseband.

Terminiert wenn Endzustand erreicht und Einleseband nicht verschiebbar ist.

$M=(Q, \Sigma, \Gamma, \Delta, q_0, F)$ mit:

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

Δ 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

5.1 Entscheidbar

5.2 Unentscheidbar

5.3 Semi-entscheidbar

6 Berechenbarkeit

7 Komplexität