Automatentheorie und formale Sprachen
rechtslineare Grammatiken

Dozentin: Wiebke Petersen

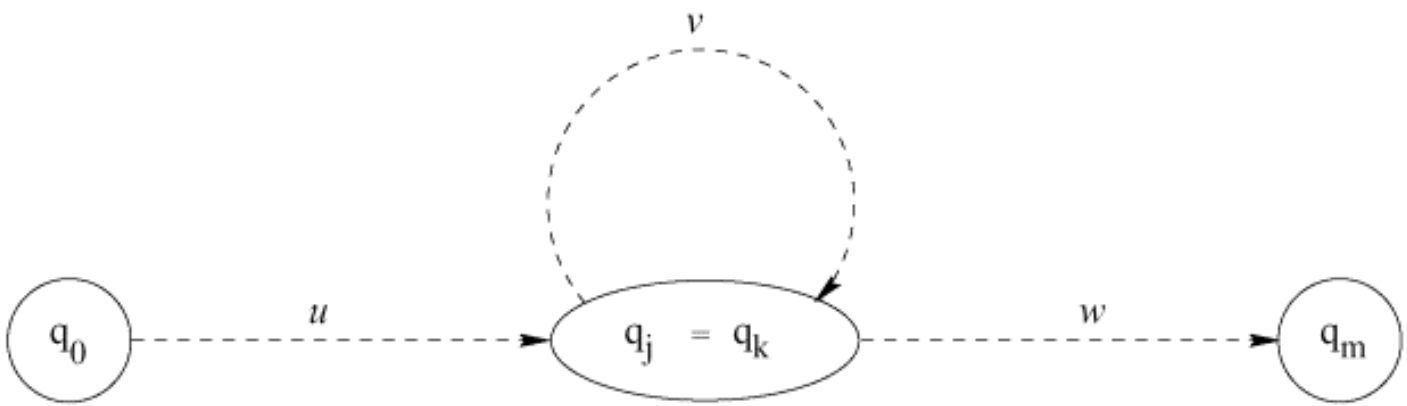
17.6.2009

Pumping lemma for regular languages

Lemma (Pumping-Lemma)

If L is an infinite regular language over Σ, then there exists a number $n \in \mathbb{N}$ such that each word $z \in L$ with $|z| \geq n$ can be decomposed into $z = uvw$ with $u, v, w \in \Sigma^*$ and $v \neq \epsilon$ such that $uv^i w \in L$ for any $i \geq 0$.

proof sketch:



zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a, b\}^* \mid w \text{ enthält eine ungerade Anzahl von } b's\}$.

zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a, b\}^* \mid w \text{ enthält eine ungerade Anzahl von } b's\}$.
   L_1 ist regulär (L_1 akzeptierenden Automaten / L_1 erzeugende rechtslineare Grammatik / regulärer Ausdruck).

zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl von $b's\}$.
   L_1 ist regulär (L_1 akzeptierenden Automaten / L_1 erzeugende rechtslineare Grammatik / regulärer Ausdruck).
2. $L_2 = \{w \in \{a,b\}^* \mid w$ enthält die gleiche Anzahl von $b's$ und $a's\}$.

zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a,b\}^* \mid w \text{ enthält eine ungerade Anzahl von } b's\}$.
   L_1 ist regulär (L_1 akzeptierenden Automaten / L_1 erzeugende rechtslineare Grammatik / regulärer Ausdruck).
2. $L_2 = \{w \in \{a,b\}^* \mid w \text{ enthält die gleiche Anzahl von } b's \text{ und } a's\}$.
   L_2 ist nicht regulär (Schnitt mit regulärer Sprache $L(a^*b^*)$ und Pumping-Lemma), obwohl für alle $v \in L_2$ auch $v^i \in L_2$.

zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a,b\}^* \mid w \text{ enthält eine ungerade Anzahl von } b's\}$.
   L_1 ist regulär (L_1 akzeptierenden Automaten / L_1 erzeugende rechtslineare Grammatik / regulärer Ausdruck).
2. $L_2 = \{w \in \{a,b\}^* \mid w \text{ enthält die gleiche Anzahl von } b's \text{ und } a's\}$.
   L_2 ist nicht regulär (Schnitt mit regulärer Sprache $L(a^*b^*)$ und Pumping-Lemma), obwohl für alle $v \in L_2$ auch $v^i \in L_2$.
3. w^R ist das Wort w in umgekehrter Reihenfolge (ww^R ist ein Palindrom).
   $L_3 = \{ww^R \mid w \in \{a,b\}^*\}$.

zu den Hausaufgaben

Welche Aussagen kann man mit Hilfe der Abschlußeigenschaften der regulären Sprachen und dem Pumping-Lemma über die Komplexität folgender formaler Sprachen machen:

1. $L_1 = \{w \in \{a,b\}^* \mid w \text{ enthält eine ungerade Anzahl von } b's\}$.
   L_1 ist regulär (L_1 akzeptierenden Automaten / L_1 erzeugende rechtslineare Grammatik / regulärer Ausdruck).
2. $L_2 = \{w \in \{a,b\}^* \mid w \text{ enthält die gleiche Anzahl von } b's \text{ und } a's\}$.
   L_2 ist nicht regulär (Schnitt mit regulärer Sprache $L(a^*b^*)$ und Pumping-Lemma), obwohl für alle $v \in L_2$ auch $v^i \in L_2$.
3. w^R ist das Wort w in umgekehrter Reihenfolge (ww^R ist ein Palindrom).
   $L_3 = \{ww^R \mid w \in \{a,b\}^*\}$.
   L_3 ist nicht regulär (Schnitt mit regulärer Sprache $L(a^*bba^*)$ und Pumping-Lemma).

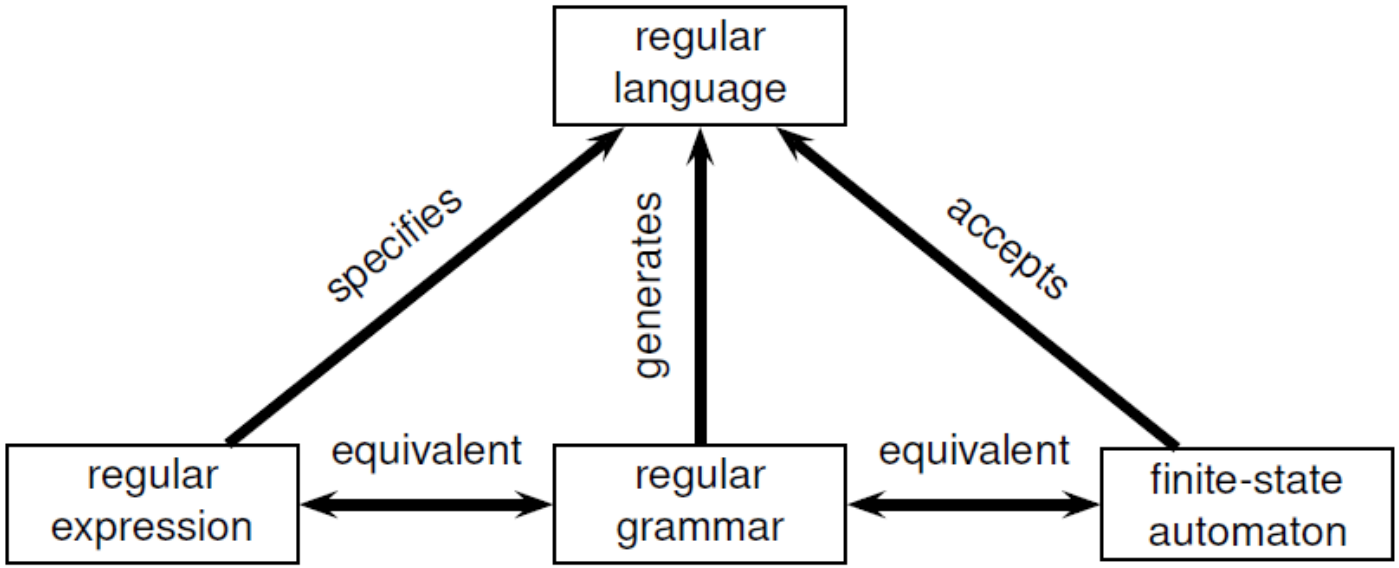
Intuitive rules for regular languages

- L is regular if it is possible to check the membership of a word simply by reading it symbol for symbol while using only a finite stack.

Intuitive rules for regular languages

- L is regular if it is possible to check the membership of a word simply by reading it symbol for symbol while using only a finite stack.
- Finite-state automatons are too weak for:
  - counting in $\mathbb{N}$ ("same number as");
  - recognizing a pattern of arbitrary length ("palindrome");
  - expressions with brackets of arbitrary depth.

Summary: regular languages



Formale Grammatik

Definition

Eine formale Grammatik ist ein 4-Tupel $G = (N, T, S, P)$ aus

- *einem Alphabet von Terminalsymbolen T (häufig auch Σ)*
- *einem Alphabet von Nichtterminalsymbolen N mit $N \cap T = \emptyset$*
- *einem Startsymbol $S \in N$*
- *einer Menge von Regeln/Produktionen*
  $P \subseteq \{\langle \alpha, \beta \rangle \mid \alpha, \beta \in (N \cup T)^* \ und \ \alpha \notin T^*\}$.

Für eine Regel $\langle \alpha, \beta \rangle$ schreiben wir auch $\alpha \rightarrow \beta$.
Formale Grammatiken werden auch Typ0- oder allgemeine Regelgrammatiken genannt.

S	$\rightarrow$	NP VP	VP	$\rightarrow$	V	NP	$\rightarrow$	D N
D	$\rightarrow$	the	N	$\rightarrow$	cat	V	$\rightarrow$	sleeps

Generiert: the cat sleeps

Formale Grammatik

Vokabular

Sei $G = (N, T, S, P)$ eine Grammatik und seien $v, w \in (T \cup N)^*$:

- v ist direkt ableitbar aus w (oder w generiert v direkt), in Zeichen $w \rightarrow v$, wenn es Zerlegungen $w = w_1 \alpha w_2$ und $v = w_1 \beta w_2$ gibt, so daß $\langle \alpha, \beta \rangle \in P$.
- v ist ableitbar aus w (oder w generiert w), in Zeichen $w \rightarrow^* v$, wenn es $w_0, w_1, \ldots w_k \in (T \cup N)^*$ gibt $(k \geq 0)$, so daß $w = w_0$, $w_k = v$ und $w_{i-1} \rightarrow w_i$ für jedes $k \geq i \geq 0$.
- $L(G) = \{w \in T^* | S \rightarrow^* w\}$ ist die von der Grammatik $G = (N, T, S, P)$ erzeugte Sprache.
- Zwei Grammatiken G_i und G_j, die dieselbe Sprache erzeugen $(L(G_i) = L(G_j))$ nennt man schwach äquivalent.

Beispiel

$$G_1 = \langle \{\text{S,NP,VP,N,V,D,N,EN}\}, \{\text{the, cat, peter, chases}\}, \text{S}, P \rangle$$

$$P = \left\{ \begin{array}{rcl} \text{S} & \rightarrow & \text{NP VP} \\ \text{NP} & \rightarrow & \text{EN} \\ \text{EN} & \rightarrow & \text{peter} \end{array} \quad \begin{array}{rcl} \text{VP} & \rightarrow & \text{V NP} \\ \text{D} & \rightarrow & \text{the} \\ \text{V} & \rightarrow & \text{chases} \end{array} \quad \begin{array}{rcl} \text{NP} & \rightarrow & \text{D N} \\ \text{N} & \rightarrow & \text{cat} \end{array} \right\}$$

Beispiel

$G_1 = \langle \{\text{S,NP,VP,N,V,D,N,EN}\}, \{\text{the, cat, peter, chases}\}, \text{S}, P \rangle$

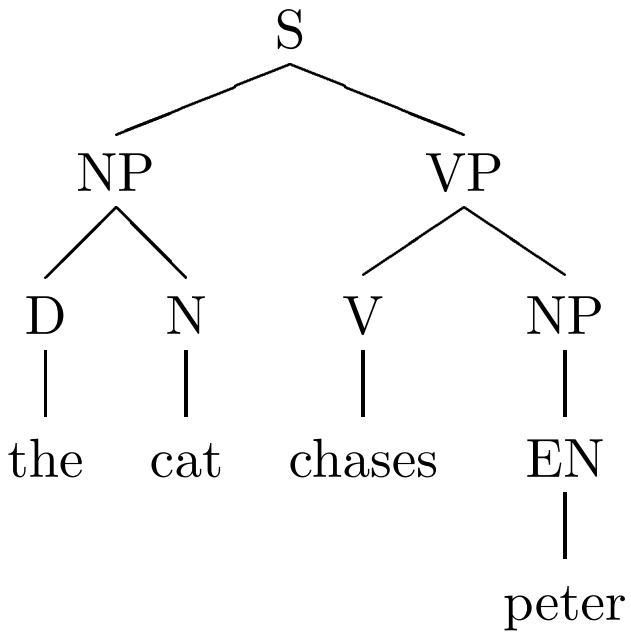
$$P = \left\{ \begin{array}{lcllcllcl} \text{S} & \rightarrow & \text{NP VP} & \text{VP} & \rightarrow & \text{V NP} & \text{NP} & \rightarrow & \text{D N} \\ \text{NP} & \rightarrow & \text{EN} & \text{D} & \rightarrow & \text{the} & \text{N} & \rightarrow & \text{cat} \\ \text{EN} & \rightarrow & \text{peter} & \text{V} & \rightarrow & \text{chases} & & & \end{array} \right\}$$

$$L(G_1) = \left\{ \begin{array}{ll} \text{the cat chases peter} & \text{peter chases the cat} \\ \text{peter chases peter} & \text{the cat chases the cat} \end{array} \right\}$$

"the cat chases peter" ist ableitbar aus S:

$$\begin{array}{llll} \text{S} & \rightarrow \text{NP VP} & \rightarrow \text{NP V NP} & \rightarrow \text{NP V EN} \\ & \rightarrow \text{NP V peter} & \rightarrow \text{NP chases peter} & \rightarrow \text{D N chases peter} \\ & \rightarrow \text{D cat chases peter} & \rightarrow \text{the cat chases peter} & \end{array}$$

Ableitungsbaum



Typ 3-Sprachen: rechts- / linkslineare Grammatiken und Sprachen

Definition

Eine Grammatik (N, T, S, P) heißt rechtslinear, wenn alle Regeln/Produktionen die folgende Form haben:
$A \rightarrow a$ oder $A \rightarrow aB$ wobei $a \in T$ und $A, B \in N$ (und gegenenfalls $S \rightarrow \epsilon$, wobei dann S nicht in einer rechten Regelseite erscheinen darf).
Ein Grammatik heißt linkslinear wenn die Regeln die Form $A \rightarrow a$ oder $A \rightarrow Ba$ (und $S \rightarrow \epsilon$, dann aber ohne S in einer rechten Regelseite) haben.
Eine durch eine rechts- bzw. linkslineare Grammatik erzeugte Sprache heißt rechts- bzw. linkslinear.

Typ 3-Sprachen: rechts- / linkslineare Grammatiken und Sprachen

Definition

Eine Grammatik (N, T, S, P) heißt rechtslinear, wenn alle Regeln/Produktionen die folgende Form haben:
$A \rightarrow a$ oder $A \rightarrow aB$ wobei $a \in T$ und $A, B \in N$ (und gegenenfalls $S \rightarrow \epsilon$, wobei dann S nicht in einer rechten Regelseite erscheinen darf).
Ein Grammatik heißt linkslinear wenn die Regeln die Form $A \rightarrow a$ oder $A \rightarrow Ba$ (und $S \rightarrow \epsilon$, dann aber ohne S in einer rechten Regelseite) haben.
Eine durch eine rechts- bzw. linkslineare Grammatik erzeugte Sprache heißt rechts- bzw. linkslinear.

Theorem

Sei L eine formale Sprache, dann sind die folgenden Aussagen äquivalent:

1. *L ist rechtslinear.*
2. *L ist linkslinear.*
3. *L ist regulär.*

Typ 3-Sprachen: rechts- / linkslineare Grammatiken und Sprachen

Definition

Eine Grammatik (N, T, S, P) heißt rechtslinear, wenn alle Regeln/Produktionen die folgende Form haben:
$A \to a$ oder $A \to aB$ wobei $a \in T$ und $A, B \in N$ (und gegenenfalls $S \to \epsilon$, wobei dann S nicht in einer rechten Regelseite erscheinen darf).
Ein Grammatik heißt linkslinear wenn die Regeln die Form $A \to a$ oder $A \to Ba$ (und $S \to \epsilon$, dann aber ohne S in einer rechten Regelseite) haben.
Eine durch eine rechts- bzw. linkslineare Grammatik erzeugte Sprache heißt rechts- bzw. linkslinear.

Theorem

Sei L eine formale Sprache, dann sind die folgenden Aussagen äquivalent:

1. *L ist rechtslinear.*
2. *L ist linkslinear.*
3. *L ist regulär.*

(Vorsicht: Nicht jede lineare Sprache ist rechtslinear)

rechtslineare Sprachen sind regulär

Satz

Zu jeder Sprache, die von einem endlichen Automaten akzeptiert wird, gibt es eine rechtslineare Grammatik, die diese Sprache erzeugt und umgekehrt.

rechtslineare Grammatik $\rightarrow$ endlicher Automat

Sei $G = (N, T, S, P)$ eine rechtslineare Grammatik, dann ist
$A = (N \cup \{\text{final}\}, T, \Delta, S, F)$ mit

- $F = \{\text{final}, S\}$ wenn $S \rightarrow \epsilon \in P$ und $F = \{\text{final}\}$ sonst.
- $(A, a, B) \in \Delta$, wenn $A \rightarrow aB \in P$ und $(A, a, \text{final}) \in \Delta$, wenn $A \rightarrow a \in P$.

ein endlicher Automat, der $L(G)$ akzeptiert.

$$S \rightarrow aA,\ S \rightarrow bB,\ S \rightarrow \epsilon,\ A \rightarrow aA,\ A \rightarrow a,\ B \rightarrow bB,\ B \rightarrow b$$

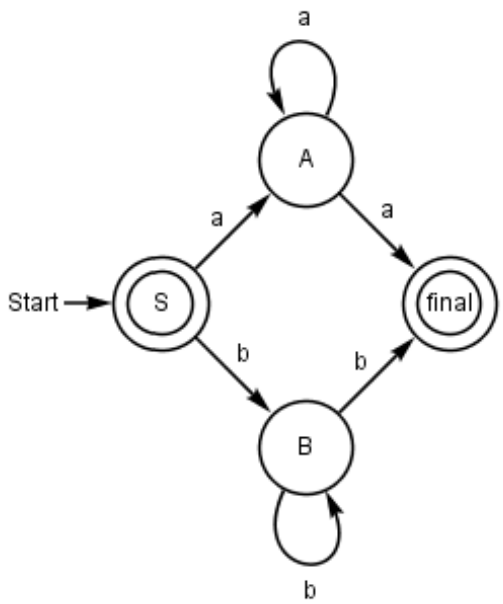
rechtslineare Grammatik $\rightarrow$ endlicher Automat

Sei $G = (N, T, S, P)$ eine rechtslineare Grammatik, dann ist
$A = (N \cup \{\text{final}\}, T, \Delta, S, F)$ mit

- $F = \{\text{final}, S\}$ wenn $S \rightarrow \epsilon \in P$ und $F = \{\text{final}\}$ sonst.
- $(A, a, B) \in \Delta$, wenn $A \rightarrow aB \in P$ und $(A, a, \text{final}) \in \Delta$, wenn $A \rightarrow a \in P$.

ein endlicher Automat, der $L(G)$ akzeptiert.

$$S \rightarrow aA,\ S \rightarrow bB,\ S \rightarrow \epsilon,\ A \rightarrow aA,\ A \rightarrow a,\ B \rightarrow bB,\ B \rightarrow b$$

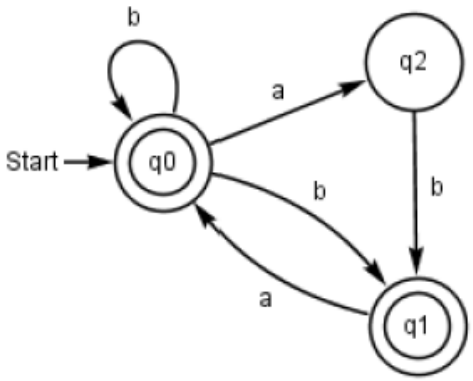


endlicher Automat $\longrightarrow$ rechtslineare Grammatik

Sei $A = (Q, \Sigma, \Delta, q_0, F)$ ein endlicher Automat ohne ϵ-Übergänge und $L(A)$, die von A akzeptierte Sprache, dann ist $G = (N, T, P, S)$ mit

- $N = Q,\ T = \Sigma,\ S = q_0$
- $q_i \longrightarrow aq_j \in P$, wenn $(q_i, a, q_j) \in \Delta$ und $q_i \longrightarrow a \in P$, wenn $(q_i, a, q_j) \in \Delta$ und $q_j \in F$.

eine rechtslineare Grammatik, die $A(L) \setminus \{\epsilon\}$ generiert.
Aus G kann relativ einfach eine Grammatik gebildet werden, die $A(L)$ generiert.
(Frage: wie?)

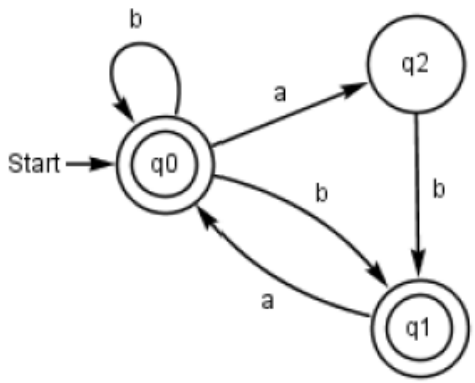


endlicher Automat $\rightarrow$ rechtslineare Grammatik

Sei $A = (Q, \Sigma, \Delta, q_0, F)$ ein endlicher Automat ohne ϵ-Übergänge und $L(A)$, die von A akzeptierte Sprache, dann ist $G = (N, T, P, S)$ mit

- $N = Q,\ T = \Sigma,\ S = q_0$
- $q_i \rightarrow aq_j \in P$, wenn $(q_i, a, q_j) \in \Delta$ und $q_i \rightarrow a \in P$, wenn $(q_i, a, q_j) \in \Delta$ und $q_j \in F$.

eine rechtslineare Grammatik, die $A(L) \setminus \{\epsilon\}$ generiert.
Aus G kann relativ einfach eine Grammatik gebildet werden, die $A(L)$ generiert.
(Frage: wie?)



$q_0 \rightarrow bq_0,\ q_0 \rightarrow b,\ q_0 \rightarrow aq_2,\ q_0 \rightarrow bq_1,\ q_2 \rightarrow bq_1,\ q_2 \rightarrow b,\ q_1 \rightarrow aq_0,\ q_1 \rightarrow a$

Hausaufgaben

Exercise 1

1. *Geben sie eine rechtslineare Grammatik zu den folgenden Sprachen an:*
   - *L ist die Sprache über dem Alphabet $\{a, b\}$, die aus allen nichtleeren Wörtern besteht, in denen auf jedes a unmittelbar ein b folgt.*
   - *L ist die Sprache über dem Alphabet $\{a, b\}$, die aus allen Wörtern besteht, in denen auf jedes a unmittelbar ein b folgt.*
   - $L(a^*b^*)$
   - *L ist die Sprache über dem Alphabet $\{a, b\}$, die aus allen Wörtern besteht, die eine gerade Anzahl von a's enthalten.*
   - $L(ab^* + aa^*)$.
2. *Sei A ein endlicher Automat. Beschreiben sie, wie man zu einer rechtslinearen Grammatik, die $L(A) \setminus \{\epsilon\}$ generiert, eine Grammatik bildet, die $L(A)$ generiert.*

Gruppenarbeit

Gruppe 1: Führen sie das in Klabunde beschriebene Verfahren zur Konstruktion eines deterministischen endlichen Automaten aus einem nichtdeterministischen Automaten an einem kleinen Beispiel im Detail durch.

Gruppe 2: Beschreiben sie an 2-3 Beispielen, wie man effizient einen deterministischen Automaten aus einem nichtdeterministischen konstruiert.

Gruppe 3: Beschreiben sie an 2-3 Beispielen, wie man ϵ-Übergänge eliminiert.

Gruppe 4: Beschreiben sie an Beispielen, wie man endliche Automaten zur Konkatenation zweier regulärer Sprachen und zur Vereinigung und zur Schnittmenge zweier Sprachen bildet, und wie man endliche Automaten zum Kleenschen Stern einer regulären Sprache und zum Komplement einer regulären Sprache konstruiert.