Theoretische Informatik

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer

Universität Freiburg
Institut für Informatik
Sommersemester 2026

Übungsblatt 7 – Lösungen

Abgabe: Dienstag, 16.06.2026, 16:00 Uhr

Aufgabe 7.1 (ε-Eliminierung; 5 Punkte)
Gegeben sei die Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit

$$\Sigma = \{a, b, c\}, \quad N = \{S, A, B, C, T, Y_a, Y_b, Y_c\} \quad \text{und}$$

$$P = \left\{ \begin{array}{l} S \to Y_a T \mid AB, \\ T \to BC, \\ A \to AY_a \mid CY_c, \\ B \to CY_b \mid \varepsilon, \\ C \to B, \\ Y_a \to a, \\ Y_b \to b, \\ Y_c \to c \end{array} \right\}$$

(a) Berechnen Sie mit dem Verfahren aus der Vorlesung die Menge der Nichtterminale, aus denen sich ε ableiten lässt:

$$\text{Nullable}(\mathcal{G}) = \{X \in N \mid X \vdash_{\mathcal{G}}^{*} \varepsilon\}$$

Geben Sie dabei in jedem Schritt die Zwischenergebnisse M_i an.

(b) Wenden Sie den Algorithmus Del aus der Vorlesung an, um $\mathcal{G}$ in eine äquivalente ε-freie Grammatik zu transformieren.

Beachten Sie, dass die gegebene Grammatik $\mathcal{G}$ bereits separiert ist und für alle Regeln $X \to \alpha$ gilt, dass $|\alpha| \leq 2$. Sie müssen die Algorithmen Sep und Bin also *nicht* als Subroutine anwenden.

Lösung:

(a)

$$M_0 = \emptyset \tag{1}$$
$$M_1 = \{B\} \tag{2}$$
$$M_2 = \{B, C\} \tag{3}$$
$$M_3 = \{B, C, T\} \tag{4}$$
$$M_4 = \{B, C, T\} \tag{5}$$

Somit gilt $\text{Nullable}(\mathcal{G}) = \{B, C, T\}$.

Bemerkung: Der Algorithmus zur Bestimmung von $\text{Nullable}(\mathcal{G})$ *ist ein Fixpunktalgorithmus. Erst aus* $M_3 = M_4$ *folgt, dass keine weiteren Nichtterminale hinzukommen und der Algorithmus terminiert.*

(b) Wir erhalten $\mathcal{G}_{\text{DEL}} = (\Sigma, N_{\text{DEL}}, P_{\text{DEL}}, S')$ mit $N_{\text{DEL}} = N \cup \{S'\}$ und

$$P_{\text{DEL}} = P \setminus \{B \to \varepsilon\} \cup \left\{ \begin{array}{l} S' \to S \\ S \to Y_a \mid A \\ T \to B \mid C \\ A \to Y_c \\ B \to Y_b \end{array} \right\}$$

Bemerkung: Da $S \notin \text{Nullable}(\mathcal{G})$*, fügen wir keine Regel* $S' \to \varepsilon$ *hinzu.*

Aufgabe 7.2 (CYK; 5 Punkte)
Gegeben sei die Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ mit

$$\Sigma = \{0, 1\}, \quad N = \{S, A, B, C\} \quad \text{und} \quad P = \left\{ \begin{array}{ll} S \to & AB \mid BC, \\ A \to & BA \mid 0, \\ B \to & CC \mid 1, \\ C \to & AB \mid 0 \end{array} \right\}$$

Bestimmen Sie mit dem CYK-Algorithmus für jedes der folgenden Wörter, ob es in $L(\mathcal{G})$ liegt:

(a) 110100

(b) 1001

Geben Sie jeweils Ihre Antwort sowie die zugehörige Matrix M an. Markieren Sie außerdem die Reihenfolge, in der die nichtleeren Zelleneinträge berechnet wurden, zum Beispiel durch Nummerierung der Einträge wie im Skript.

Lösung:
Analog zur Vorlesung gibt der Index eines Nichtterminals außerhalb der Diagonale an, an welcher Stelle der jeweilige nicht-leere Zelleneintrag eingetragen wurde. Ein $\bullet$ steht für die leere Menge.

(a) Für $w =$ `110100` ergibt sich:

B	$\bullet$	A_5	C, S_8	B_{11}	A, S_{13}
1	B	A, S_1	C, S_6	B_9	A, S_{12}
	1	A, C	C, S_2	B_7	A, S_{10}
		0	B	A, S_3	$\bullet$
			1	A, C	B_4
				0	A, C
					0

Da $S \in M_{1,6}$, gilt $\texttt{110100} \in L(\mathcal{G})$.

(b) Für $w = \texttt{1001}$ ergibt sich:

B	A, S_1	$\bullet$	$\bullet$	
1	A, C	B_2	B_4	
	0	A, C	C, S_3	
		0	B	
			1	

Da $S \notin M_{1,4}$, gilt $\texttt{1001} \notin L(\mathcal{G})$.

Aufgabe 7.3 (Pumping Lemma für Kontextfreie Sprachen; 5 Punkte)
Zeigen Sie mithilfe des Pumping-Lemmas für kontextfreie Sprachen, dass die folgende Sprache nicht kontextfrei ist:

$$L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^\ell \mid n > m \text{ und } n > \ell\}$$

Lösung:
Angenommen, L ist kontextfrei. Dann gibt es nach dem Pumping Lemma für kontextfreie Sprachen eine Pumping-Länge $p \in \mathbb{N}$, so dass jedes Wort $z \in L$ mit $|z| \geq p$ eine Zerlegung $z = uvwxy$ mit

$$|vwx| \leq p, \qquad |vx| \geq 1$$

besitzt, so dass für alle $i \geq 0$ gilt: $uv^iwx^iy \in L$.

Wir wählen

$$z = \mathtt{a}^{p+1}\mathtt{b}^p\mathtt{c}^p \in L \quad \text{ mit } |z| = 3p + 1 > p$$

Sei nun $z = uvwxy$ eine beliebige Zerlegung mit $|vwx| \leq p$ und $|vx| \geq 1$. Da der Block $\mathtt{b}^p$ die Länge p hat, kann das Teilwort vwx nicht gleichzeitig ein $\mathtt{a}$ und ein $\mathtt{c}$ enthalten.

Wir unterscheiden die folgenden beiden Fälle.

(a) Es gilt $\#_{\mathtt{a}}(vx) \geq 1$.

Wegen $\#_{\mathtt{a}}(vwx) \geq 1$ und $|vwx| \leq p$ gilt nach obiger Beobachtung $\#_{\mathtt{c}}(vx) = 0$. Für $i = 0$ folgt daher

$$\#_{\mathtt{a}}(uwy) = p + 1 - \#_{\mathtt{a}}(vx) \leq p = \#_{\mathtt{c}}(uwy).$$

Somit ist die für Wörter aus L notwendige Ungleichung $\#_{\mathtt{a}}(uwy) > \#_{\mathtt{c}}(uwy)$ verletzt, also $uwy \notin L$.

(b) Es gilt $\#_{\mathtt{a}}(vx) = 0$.

Aus $|vx| \geq 1$ folgt

$$\#_{\mathtt{b}}(vx) + \#_{\mathtt{c}}(vx) \geq 1.$$

Für $i = 2$ gelten

$$\#_{\mathtt{a}}(uv^2wx^2y) = p + 1,$$

sowie

$$\#_{\mathtt{b}}(uv^2wx^2y) = p + \#_{\mathtt{b}}(vx), \qquad \#_{\mathtt{c}}(uv^2wx^2y) = p + \#_{\mathtt{c}}(vx).$$

Mindestens eine dieser beiden Anzahlen ist daher mindestens $p+1$. Folglich ist mindestens eine der notwendigen Ungleichungen $\#_{\mathtt{a}}(uv^2wx^2y) > \#_{\mathtt{b}}(uv^2wx^2y)$ und $\#_{\mathtt{a}}(uv^2wx^2y) > \#_{\mathtt{c}}(uv^2wx^2y)$ verletzt. Also gilt auch hier $uv^2wx^2y \notin L$.

In beiden Fällen erhalten wir einen Widerspruch zum Pumping Lemma. Folglich ist L nicht kontextfrei.

□

Bemerkung: In der Fallunterscheidung verwenden wir die Implikationen

$$\neg(\#_{\mathtt{a}}(w) > \#_{\mathtt{b}}(w)) \implies w \notin L \qquad \textit{und} \qquad \neg(\#_{\mathtt{a}}(w) > \#_{\mathtt{c}}(w)) \implies w \notin L.$$

Daher muss nicht zusätzlich überprüft werden, ob das gepumpte Wort weiterhin die Blockform $\mathtt{a}^*\mathtt{b}^*\mathtt{c}^*$ *besitzt. Geht diese durch das Pumpen verloren, so ist dies einfach ein weiterer Grund dafür, dass das gepumpte Wort nicht in L liegt.*

Aufgabe 7.4 (Typ-3-Sprachen; 5 Punkte)

In der Vorlesung haben Sie sowohl reguläre Sprachen als auch Typ-3-Sprachen kennengelernt. Für eine Sprache $L \subseteq \Sigma^*$ gilt

$$L \in \mathit{REG} \iff L \text{ ist eine Typ-3-Sprache.}$$

In dieser Aufgabe sollen Sie diese Äquivalenz zeigen, indem Sie Konstruktionen in beide Richtungen angeben.

(a) ($\implies$): Sei $L \in \mathit{REG}$ und sei $\mathcal{A}$ ein DEA mit $L(\mathcal{A}) = L$. Konstruieren Sie eine Typ-3-Grammatik $\mathcal{G}$ mit $L(\mathcal{G}) = L$.

(b) ($\Longleftarrow$): Sei L eine Typ-3-Sprache und sei $\mathcal{G}$ eine Typ-3-Grammatik mit $L(\mathcal{G}) = L$. Konstruieren Sie einen NEA $\mathcal{N}$ mit $L(\mathcal{N}) = L$.

Hinweis: Es genügt, die jeweilige Konstruktion anzugeben. Sie müssen *nicht* formal beweisen, dass die Konstruktionen korrekt sind.

Lösung:

Wir geben Konstruktionen in beide Richtungen an.

(a) ($\Longrightarrow$)

Sei $\mathcal{A} = (\Sigma, Q, \delta, q^{\text{init}}, F)$ ein DEA mit $L(\mathcal{A}) = L$. Wir konstruieren die Typ-3-Grammatik $\mathcal{G} = (\Sigma, N, P, S)$ durch

$$N = Q, \qquad S = q^{\text{init}}$$

und

$$P = \{q \to aq' \mid q, q' \in Q,\ a \in \Sigma,\ \delta(q, a) = q'\} \cup \{q \to \varepsilon \mid q \in F\}.$$

(b) ($\Longleftarrow$)

Sei $\mathcal{G} = (\Sigma, N, P, S)$ eine Typ-3-Grammatik mit $L(\mathcal{G}) = L$. Durch Anwenden von Bin erhalten wir eine äquivalente Grammatik, in der jede Regel eine der folgenden Formen hat:

$A \to \varepsilon$	für $A \in N$,
$A \to a$	für $A \in N,\ a \in \Sigma$,
$A \to a_1 a_2$	für $A \in N,\ a_1, a_2 \in \Sigma$,
$A \to aB$	für $A \in N,\ a \in \Sigma,\ B \in N$.

Wir führen nun die folgenden Modifikationen durch.

- Wir fügen eine frische Variable Y_ε und die Regel $Y_\varepsilon \to \varepsilon$ hinzu.
- Wir löschen jede Regel der zweiten Form und führen stattdessen die Regel $A \to aY_\varepsilon$ ein.
- Wir löschen jede Regel der dritten Form und führen stattdessen eine frische Variable Y_{a_2} und die Regeln $A \to a_1 Y_{a_2}$ und $Y_{a_2} \to a_2 Y_\varepsilon$ ein.

Die resultierende Grammatik $\mathcal{G}' = (\Sigma, N', P', S)$ ist äquivalent zu $\mathcal{G}$.

Aus $\mathcal{G}'$ konstruieren wir den NEA

$$\mathcal{N} = (\Sigma, Q, \delta, q^{\text{init}}, F)$$

mit

$$Q = N', \qquad q^{\text{init}} = S, \qquad F = \{A \in N' \mid A \to \varepsilon \in P'\}.$$

Die Übergangsfunktion ist durch

$$B \in \delta(A, a) \quad \Longleftrightarrow \quad A \to aB \in P'$$

definiert. Dann simuliert jeder Automatenschritt genau eine Ableitungsregel der Grammatik, und es gilt $L(\mathcal{N}) = L(\mathcal{G}') = L$.