

FORMALE SYSTEME

ÜBUNG 6

Eric Kunze

`eric.kunze@tu-dresden.de`

TU Dresden, 26. November 2021

letzte Änderung:
26.11.2021, 13:04

Aufgabe 1:

Pumping-Lemma

NICHTREGULARITÄT DURCH PUMPEN

Idee:

- ▶ Jeder DFA hat nur endlich viele Zustände n
- ▶ Aber manche reguläre Sprachen enthalten beliebig lange Wörter

Wie kann ein DFA Wörter mit mehr als n Zeichen akzeptieren?

- ▶ Dann muss der DFA beim Einlesen einen Zustand mehr als einmal besuchen
- ▶ Dafür muss es in den Zustandsübergängen eine Schleife geben
- ▶ Diese Schleife kann man aber auch mehr als einmal durchlaufen

Jedes akzeptierte Wort mit $\geq n$ Zeichen hat einen Teil, den man beliebig oft wiederholen – „aufpumpen“ – kann.

DAS PUMPING-LEMMA

Satz (Pumping-Lemma): Für jede reguläre Sprache \mathbf{L} gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvw$ mit $|v| \geq 1$ und $|uv| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^k w \in \mathbf{L}$

Beweis: Sei \mathcal{M} ein DFA für \mathbf{L} mit $|Q|$ Zuständen. Wir wählen $n = |Q| + 1$.

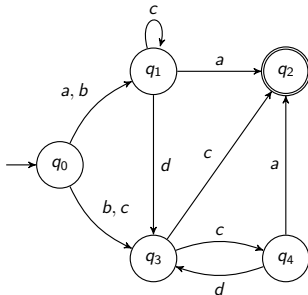
Ein akzeptierender Lauf für ein beliebiges Wort z mit $|z| = \ell \geq n$ muss in den ersten n Schritten einen Zustand p zweimal besuchen (sagen wir: nach i und j Schritten), hat also die Form:

$$q_0 \xrightarrow{z_1} q_1 \xrightarrow{z_2} \dots \xrightarrow{z_{i-1}} q_{i-1} \xrightarrow{z_i} p \xrightarrow{z_{i+1}} q_{i+1} \xrightarrow{z_{i+2}} \dots \xrightarrow{z_{j-1}} q_{j-1} \xrightarrow{z_j} p \xrightarrow{z_{j+1}} q_{j+1} \xrightarrow{z_{j+2}} \dots \xrightarrow{z_\ell} q_\ell$$

Die gesuchte Zerlegung ist $u = z_1 \cdots z_i$, $v = z_{i+1} \cdots z_j$, $w = z_{j+1} \cdots z_\ell$.

Der Lauf $(q_0 \dots q_{i-1} p)(q_{i+1} \dots q_{j-1} p)^k (q_{j+1} \dots q_\ell)$ akzeptiert $uv^k w$. □

AUFGABE 1



- a) Geben Sie für jedes $z \in \{bc, adc, cda, bcdbc, acdc\}$ alle Zerlegungen $z = uvw$ mit $u, w \in \Sigma^*$, $v \in \Sigma^+$ an, sodass für alle $k \geq 0$ gilt: $uv^k w \in L(\mathcal{M})$. Begründen Sie Ihre Antworten.
- b) Ermitteln Sie eine Zahl $n \in \mathbb{N}$, sodass für alle $z \in L(\mathcal{M})$ mit $|z| \geq n$ gilt, dass eine Zerlegung $z = uvw$ mit $u, w \in \Sigma^*$, $v \in \Sigma^+$ und $|uv| \leq n$ existiert, sodass für alle $k \geq 0$ gilt: $uv^k w \in L(\mathcal{M})$.

- a) Idee: Aufpumpen entspricht Zyklen/Schleifen in \mathcal{M}
 \rightsquigarrow möglich durch Schleife in q_1 oder Zyklus $q_3 \rightarrow q_4 \rightarrow q_3$
- ▷ Für $z \in \{bc, adc\}$ existiert keine Zerlegung (beachte, dass stets auch $k = 0$ zulässig sein muss)
 - ▷ $z = cda \notin L(\mathcal{M})$
 - ▷ $z = bc dc \rightsquigarrow b \mid c \mid dc$ oder $b \mid cd \mid c$ oder $bc \mid dc \mid \varepsilon$
 - ▷ $z = ac dc \rightsquigarrow a \mid c \mid dc$
- b) Laut Beweis des Pumping-Lemmas ist $n = |Q| + 1 = 6$ zulässig. Tatsächlich reicht auch $n = 5$, da ein Lauf mit i Übergängen automatisch $i + 1$ Zustände besucht. Auch $n = 4$ ist ausreichend, wenn man sich überlegt wie Wörter der Länge 4 aussehen dürfen:
- ▷ Weg über q_3 : nutze Zyklus $q_3 \rightarrow q_4 \rightarrow q_3$
 - ▷ Weg über q_1 und q_3 – Variante 1: nutze Loop in q_1
 - ▷ Weg über q_1 und q_3 – Variante 2: nutze Zyklus $q_3 \rightarrow q_4 \rightarrow q_3$ (evtl. mit Start in q_1 , d.h.
 $q_1 \rightarrow q_3 \rightarrow q_4 \rightarrow q_3 \rightarrow \dots \rightarrow q_4 \rightarrow q_2$)

Aufgabe 2:

Regularität von Sprachen

BEWEIS VON NICHTREGULARITÄT

Satz (Myhill & Nerode): Eine Sprache L ist genau dann regulär, wenn \simeq_L endlich viele Äquivalenzklassen hat.

Satz: Wenn L_1 und L_2 regulär sind, dann auch $L_1 \cap L_2$, $L_1 \cup L_2$, L_1^* und \bar{L}_1 .

Satz (Pumping-Lemma): Für jede reguläre Sprache L gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $x \in L$ mit $|x| \geq n$
gibt es eine Zerlegung $x = uvw$ mit $|v| \geq 1$ und $|uv| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^k w \in L$

AUFGABE 2

Gegeben ist das Alphabet $\Sigma = \{a, b\}$. Welche der folgenden Sprachen L_j über Σ mit $1 \leq j \leq 2$ ist regulär? Beweisen Sie Ihre jeweilige Antwort.

a) $L_1 = \{a^i b^i : 1 \leq i \leq 15\}$

b) $L_2 = \{a^n b^m a^{n \cdot m} : n, m \geq 0\}$

- a) Die Sprache L_1 ist endlich, da i eine obere Grenze hat. Jede endliche Sprache ist regulär.
- b) L_2 ist nicht regulär — Pumping-Lemma: Angenommen L_2 sei regulär. Dann existiert nach dem Pumping-Lemma ein $n \geq 0$, sodass jedes Wort $x \in L_2$ mit $|x| \geq n$ gepumpt werden kann. Insbesondere muss dies auch für das Wort $x = a^n b^1 a^{n-1}$ gelten. Dementsprechend muss es eine Zerlegung

$$x = uvw \quad \text{mit } |uv| \leq n \text{ und } |v| \geq 1$$

geben. Wegen $|uv| \leq n$ muss $uv = a^\ell$ mit $1 \leq \ell \leq n$ gelten (d.h. uv liegt in den ersten a 's). Ein Teil der a 's muss dem v zugeschrieben werden, d.h. es gilt $v = a^h$ mit $1 \leq h \leq \ell$. Damit können wir nun pumpen, insbesondere mit $k = 2$:

$$uv^2w = a^{\ell-h} a^{2h} a^{n-\ell} b a^n = a^{\ell-h+2h+n-\ell} b a^n = a^{n+h} b a^n \notin L_2$$

im Widerspruch zur Aussage des Pumping-Lemmas, was $uv^2w \in L_2$ sichern würde. Damit kann die Annahme der Regularität nicht richtig gewesen sein und L_2 ist nicht regulär.

Aufgabe 3:

Wiederholung

AUFGABE 3

Lösung

Beweisen oder widerlegen Sie unter Verwendung von Resultaten aus der Vorlesung folgende Aussagen.

- a) ✗ Für die Grammatik $G = (\{S, X, Y, Z\}, \{a, b\}, \{S \rightarrow Y, X \rightarrow b, Y \rightarrow aYYb, aY \rightarrow aZ, ZY \rightarrow ZX, Z \rightarrow a\}, S)$ gilt: $abab \in L(G)$.
- b) ✓ Kann eine Sprache L von einem DFA erkannt werden, so gibt es auch einen ε -NFA \mathcal{M} mit $L(\mathcal{M}) = L$.
- c) ✓ Für jeden NFA \mathcal{M} mit Wortübergängen gibt es einen äquivalenten NFA.
- d) ✓ Es gibt eine reguläre Sprache, für welche die Anzahl der Äquivalenzklassen der *Nerode*-Rechtskongruenz endlich ist.
- e) ✓ Wenn es für eine Sprache L ein $n \in \mathbb{N}$ gibt, so dass die *Nerode*-Rechtskongruenz \simeq_L höchstens n Äquivalenzklassen hat, so kann L von einem DFA erkannt werden.
- f) ✓ Für jede Sprache L gilt: $L = \bigcup_{u \in L} [u]_{\simeq_L}$.

Aufgabe 4

Chomsky-Normalform

ELIMINIEREN VON ε -REGELN

Eingabe: CFG $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: ε -freie CFG $G' = \langle V', \Sigma, P', S' \rangle$ mit $L(G') = L(G)$

- ▶ Initialisiere $P' := P$ und $V' := V$
- ▶ Berechne $V_\varepsilon = \{A \in V \mid A \Rightarrow^* \varepsilon\}$
- ▶ Entferne alle ε -Regeln aus P'
- ▶ Solange es in P' eine Regel $B \rightarrow xAy$ gibt, mit

$$A \in V_\varepsilon \quad |x| + |y| \geq 1 \quad B \rightarrow xy \notin P'$$

wähle eine solche Regel und setze $P' := P' \cup \{B \rightarrow xy\}$

- ▶ Falls $S \in V_\varepsilon$ dann definiere ein neues Startsymbol $S' \notin V$, setze $V' := V' \cup \{S'\}$ und $P' := P' \cup \{S' \rightarrow S, S' \rightarrow \varepsilon\}$. Falls $S \notin V_\varepsilon$, dann verwenden wir einfach $S' := S$ als Startsymbol.

ELIMINIERUNG VON KETTENREGELN

Eine **Kettenregel** ist eine Regel der Form $A \rightarrow B$.

Sei $G = \langle V, \Sigma, P, S \rangle$ ε -frei. Eine äquivalente Grammatik ohne Kettenregeln ist gegeben durch $G' = \langle V, \Sigma, P', S \rangle$:

Eliminieren von Kettenregeln:

$E(A)$... Menge aller $B \in V$, die man von $A \in V$ aus über Kettenregeln erreichen kann:

(1) $A \in E(A)$

(2) Falls $B \in E(A)$ und $B \rightarrow B' \in P$ mit $B' \in V$ dann $B' \in E(A)$.
Wiederhole.

$$\Rightarrow P' = \bigcup_{A \in V} \left\{ A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A) \right\}$$

DIE CHOMSKY-NORMALFORM

Eine kontextfreie Grammatik $G = \langle V, \Sigma, P, S \rangle$ ist in **Chomsky-Normalform (CNF)**, wenn alle ihre Produktionsregeln eine der beiden folgenden Formen haben:

$$A \rightarrow BC \quad (\text{mit } B, C \in V) \quad \text{oder} \quad A \rightarrow c \quad (\text{mit } c \in \Sigma)$$

Umwandlung in CNF:

- (1) Eliminierung von ε -Regeln
- (2) Eliminierung von Kettenregeln

(3) Extrahiere Regeln der Form $A \rightarrow c$, so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten.

- ▷ für jedes Symbol $a \in \Sigma$:
neue Variable V_a mit Regel $V_a \rightarrow a$
- ▷ für Regeln $A \rightarrow w$ mit $|w| > 1$:
ersetze jedes Vorkommen von $a \in \Sigma$ in w durch V_a

(4) Reduziere Regeln der Form $A \rightarrow B_1 \cdots B_n$ auf $n = 2$

Für jede Produktionsregel $A \rightarrow B_1 \cdots B_n$ mit $n > 2$:

- ▷ Führe $n - 2$ neue Variablen C_1, \dots, C_{n-2} ein
- ▷ Ersetze die Regel durch neue Regeln:

$$\begin{aligned} A &\rightarrow B_1 C_1 \\ C_1 &\rightarrow B_2 C_2 \\ &\vdots \\ C_{n-3} &\rightarrow B_{n-2} C_{n-2} \\ C_{n-2} &\rightarrow B_{n-1} B_n \end{aligned}$$

AUFGABE 4

Betrachten Sie die Grammatik $G_0 = \langle V, \Sigma, P, S \rangle$ mit $V = \{S, T, U, V, R\}$, $\Sigma = \{a, b\}$ und

$$P = \left\{ \begin{array}{llll} S \rightarrow \varepsilon, & S \rightarrow aSb, & S \rightarrow T, & S \rightarrow R, \\ T \rightarrow bbT, & T \rightarrow U, & U \rightarrow aaU, & U \rightarrow bbT, \\ V \rightarrow bSa, & R \rightarrow \varepsilon, & R \rightarrow bSa & \end{array} \right\}$$

- a) Konstruieren Sie eine Grammatik G_1 , die keine Regeln der Form $A \rightarrow \varepsilon$ für $A \in V$ enthält. Erweitern Sie dazu, wenn nötig, die Grammatik G_0 um ein neues Startsymbol S' und entsprechende Regeln.
- b) Geben Sie zu G_1 eine äquivalente Grammatik G_2 an, die keine Kettenregeln, also Produktionen der Form $A \rightarrow B$ mit Nichtterminalsymbolen A, B , enthält.
- c) Geben Sie eine Grammatik G_3 in Chomsky-Normalform an mit $L(G_3) = L(G_2) \setminus \{\varepsilon\}$.

Zur Vereinfachung entfernen wir nicht erreichbare Symbole (V) und nicht terminierende Symbole (T, U).¹ Damit sieht die Regelmenge in Kurznotation wie folgt aus:

$$P = \left\{ \begin{array}{l} S \rightarrow \varepsilon \mid aSb \mid R \\ R \rightarrow \varepsilon \mid bSa \end{array} \right\}$$

a) **Eliminieren der ε -Regeln** laut Algorithmus: $V_\varepsilon = \{S, A\}$

$$P_1 = \left\{ \begin{array}{l} S \rightarrow aSb \mid R \mid ab \\ R \rightarrow bSa \mid ba \\ S' \rightarrow \varepsilon \mid S \end{array} \right\}$$

Damit erhalten wir $G_1 = \langle V_1, \Sigma, P_1, S' \rangle$ mit $V_1 = \{S, R, S'\}$ und P_1 von oben.

¹Diese Vereinfachung ist nicht Bestandteil des Algorithmus.

b) **Eliminieren von Kettenregeln:** Erreichbarkeitsmengen:

$$E(S) = \{S, R\}, \quad E(R) = \{R\}, \quad E(S') = \{S', S, R\}$$

$$P_{\text{neu}} = \bigcup_{A \in V} \left\{ A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A) \right\}$$

$$P_2 = \left\{ \begin{array}{ll} S \rightarrow \underbrace{aSb \mid ab}_{B=S} \mid \underbrace{bSa \mid ba}_{B=R} & A = S \\ R \rightarrow \underbrace{bSa \mid ba}_{B=R} & A = R \\ S' \rightarrow \underbrace{\varepsilon}_{B=S'} \mid \underbrace{aSb \mid ab}_{B=S} \mid \underbrace{bSa \mid ba}_{B=R} & A = S' \end{array} \right\}$$

Damit erhalten wir $G_2 = \langle V_1, \Sigma, P_2, S' \rangle$ mit $V_1 = \{S, R, S'\}$ wie bisher und P_2 von oben.

c) **Chomsky-Normalform:**

Extrahieren von Terminalen

$$\left\{ \begin{array}{lcl} S & \rightarrow & V_a \textcolor{violet}{S} V_b \mid V_a V_b \mid V_b \textcolor{blue}{S} V_a \mid V_b V_a \\ R & \rightarrow & V_b \textcolor{blue}{S} V_a \mid V_b V_a \\ S' & \rightarrow & \varepsilon \mid V_a \textcolor{violet}{S} V_b \mid V_a V_b \mid V_b \textcolor{blue}{S} V_a \mid V_b V_a \\ V_a & \rightarrow & a \\ V_b & \rightarrow & b \end{array} \right\}$$

Verkürzen der Nichtterminalsequenzen

$$P'_3 = \left\{ \begin{array}{lcl} S & \rightarrow & V_a \textcolor{violet}{C}_b \mid V_a V_b \mid V_b \textcolor{blue}{C}_a \mid V_b V_a \\ R & \rightarrow & V_b \textcolor{blue}{C}_a \mid V_b V_a \\ S' & \rightarrow & \varepsilon \mid V_a \textcolor{violet}{C}_b \mid V_a V_b \mid V_b \textcolor{blue}{C}_a \mid V_b V_a \\ V_a & \rightarrow & a \\ V_b & \rightarrow & b \\ \textcolor{blue}{C}_a & \rightarrow & S V_a \\ \textcolor{violet}{C}_b & \rightarrow & S V_b \end{array} \right\}$$

Damit ist $G_3 = \langle V_3, \Sigma, P_3, S' \rangle$ mit $V_3 = \{S', S, R, V_a, V_b, C_a, C_b\}$ und $P_3 = P'_3 \setminus \{S' \rightarrow \varepsilon\}$ in Chomsky-Normalform. Es gilt jedoch $L(G_3) = L(G_2) \setminus \{\varepsilon\}$.

Aufgabe 5

CYK-Algorithmus

gegeben: kontextfreie Grammatik G in CNF

Frage: $w = a_1 \cdots a_n \in L(G)$?

- ▶ Falls $|w| = 1$, dann ist $w \in \Sigma$ und es gilt:
 $w \in L(G)$ genau dann wenn es eine Regel $S \rightarrow w$ in G gibt
- ▶ Falls $|w| > 1$, dann ist:
 $w \in L(G)$ genau dann wenn es eine Regel $S \rightarrow AB$ und eine Zahl i gibt, so dass gilt

$$A \Rightarrow^* a_1 \cdots a_i \quad \text{und} \quad B \Rightarrow^* a_{i+1} \cdots a_n$$

Idee: Fall 2 reduziert das Problem $S \stackrel{?}{\Rightarrow}^* w$ auf zwei einfachere Probleme $A \stackrel{?}{\Rightarrow}^* a_1 \cdots a_i$ und $B \stackrel{?}{\Rightarrow}^* a_{i+1} \cdots a_n$, die man allerdings für alle Regeln $S \rightarrow AB$ und Indizes i lösen muss

CYK: PRAKTISCHE UMSETZUNG

Vorgehen: $V[i, j]$ = Menge aller A mit $A \Rightarrow^* w_{i,j}$

$\rightsquigarrow V[i, j]$ können in einer Dreiecksmatrix notiert werden

- ▶ Diagonale = Fall 1: existiert Terminalsymbolregel
- ▶ Fixiere Element \blacksquare : sei \blacktriangleleft in der gleichen Zeile ganz links und \blacktriangledown direkt unten drunter
 - ▷ wenn eine Regel $\blacklozenge \rightarrow \blacktriangleleft \blacktriangledown$, dann füge \blacklozenge zu \blacksquare hinzu
 - ▷ schiebe \blacktriangleleft nach rechts und \blacktriangledown nach unten und wiederhole

Ist am Ende das Startsymbol $S \in V[1, |w|]$, dann liegt w in der Sprache

Beispiel: Wir betrachten das Wort $w = a + b \cdot c$ der Länge $|w| = 5$.

| | | | | | |
|---|-----------|----------------------|-----------|-----------------------------------|-----------|
| a | $V[1, 1]$ | $V[1, 2]$ | $V[1, 3]$ | $V[1, 4]$ | $V[1, 5]$ |
| + | | \blacktriangleleft | $V[2, 3]$ | $\blacksquare \cup \blacklozenge$ | $V[2, 5]$ |
| b | | | $V[3, 3]$ | \blacktriangledown | $V[3, 5]$ |
| . | | | | $V[4, 4]$ | $V[4, 5]$ |
| c | | | | | $V[5, 5]$ |
| | a | + | b | . | c |

AUFGABE 5

Gegeben ist folgende Grammatik $G = (V, \Sigma, P, S)$ mit
 $V = \{S, X, M, A, B\}$, $\Sigma = \{a, b\}$ und

$$P = \left\{ \begin{array}{lll} S \rightarrow \varepsilon, & S \rightarrow AX, & S \rightarrow AB, \\ X \rightarrow MB, & & \\ M \rightarrow AB, & M \rightarrow AX, & \\ A \rightarrow a, & & \\ B \rightarrow a, & B \rightarrow b & \end{array} \right\}$$

Verwenden Sie den CYK-Algorithmus (mit der Matrix-Notation aus der Vorlesung), um für die folgenden Wörter w_i zu entscheiden, ob $w_i \in L(G)$ ist.

a) $w_1 = aaabba$

b) $w_2 = aabbbaa$

a) $w_1 = aaabba$

| | | | | | | |
|---|------|------|------|------|-------------|-------------|
| a | A, B | S, M | X | S, M | X | S, M |
| a | | A, B | S, M | X | S, M | X |
| a | | | A, B | S, M | X | \emptyset |
| b | | | | B | \emptyset | \emptyset |
| b | | | | | B | \emptyset |
| a | | | | | | A, B |
| | a | a | a | b | b | a |

 $\Rightarrow w_1 \in L(G)$ b) $w_2 = aabbba$

| | | | | | | |
|---|------|------|------|-------------|-------------|-------------|
| a | A, B | S, M | X | S, M | X | \emptyset |
| a | | A, B | S, M | X | \emptyset | \emptyset |
| b | | | B | \emptyset | \emptyset | \emptyset |
| b | | | | B | \emptyset | \emptyset |
| a | | | | | A, B | S, M |
| a | | | | | | A, B |
| | a | a | b | b | a | a |

 $\Rightarrow w_2 \notin L(G)$