

## Einführung in die Theoretische Informatik

### Sommersemester 2022 – Übungsblatt 3

- Das Übungsblatt ist in zwei Teile gegliedert: den Vorbereitungsteil, den Sie vor der Übung selbstständig bearbeiten sollen, und den Übungs-/Nachbereitungsteil, der Aufgaben enthält, die in der Übung besprochen werden und von Ihnen anschließend zur Nachbereitung verwendet werden können.
- Das ist nicht das Hausaufgabenblatt! Die Hausaufgaben finden Sie auf einem separaten Blatt.

#### Vorbereitung ( $\rightarrow$ vor der Übung selbstständig zu bearbeiten)

##### Individualaufgabe Ü3.1. (Wichtige Begriffe)

Überprüfen Sie, dass Sie die folgenden Begriffe oder Notationen korrekt definieren können.

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>• Wortproblem</li><li>• Leerheitsproblem</li><li>• Endlichkeitsproblem</li></ul> | <ul style="list-style-type: none"><li>• Äquivalenzproblem</li><li>• Ardens Lemma</li><li>• Pumping Lemma</li></ul> |
|--|--|

##### Individualaufgabe Ü3.2. (Kahoot)

Falls Sie eines der Kahoots aus der Vorlesung verpasst haben: Spielen Sie es jetzt! [Link](#). Um ein Kahoot zu starten, gehen Sie auf „Play as guest“, dann „Continue as guest“ und „Classic mode“.

##### Individualaufgabe Ü3.3. (Automata Tutor: Pumping Lemma Game)

Lösen Sie die Aufgaben ~~Ü3.3~~ (a–b) auf [Automata Tutor](#).<sup>1</sup>

**Update:** Aktuell hat AT anscheinend Probleme mit diesem Aufgabentyp. Bis wir das beheben können, haben wir die Aufgaben deaktiviert.

##### Individualaufgabe Ü3.4. (Strukturelle Induktion)

Geben Sie eine rekursive Prozedur  $empty(r)$  an, die für einen gegebenen regulären Ausdruck  $r$  entscheidet, ob  $L(r) = \emptyset$ . Für Ihre Definition sollten Sie das folgende Gerüst verwenden:

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>• <math>empty(\emptyset) =</math></li><li>• <math>empty(a) =</math></li><li>• <math>empty(\epsilon) =</math></li></ul> | <ul style="list-style-type: none"><li>• <math>empty(\alpha\beta) = empty(\alpha) \vee empty(\beta)</math></li><li>• <math>empty(\alpha \mid \beta) =</math></li><li>• <math>empty(\alpha^*) =</math></li></ul> |
|--|--|

Beweisen Sie mittels struktureller Induktion, dass Ihre Definition korrekt ist.

Zu dieser Aufgabe gibt es eine Video-Lösung: [rekursive Prozedur, strukturelle Induktion](#).

---

<sup>1</sup>Wenn Sie Automata Tutor noch nicht verwendet haben, folgen Sie erst den Schritten in Ü1.2, um sich richtig zu registrieren.

regulär  $\Rightarrow$  PL

## Übung und Nachbereitung

### Übungsaufgabe Ü3.5. (Pumping Lemma)

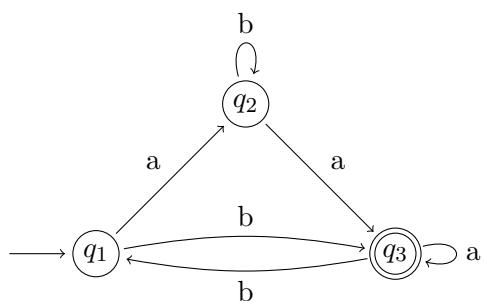
Beweisen Sie für jede der folgenden Sprachen mithilfe des Pumping Lemmas, dass sie *nicht* regulär sind.

- (a)  $L_1 = \{w \in \{0,1\}^* \mid w = w^R\}$
- (b)  $L_2 = \{w \in \{0,1\}^* \mid |w|_0 \geq |w|_1\}$
- (c)  $L_5 = \{a^{2^i} \mid i \geq 0\}$

$|w|_0 = \text{"Anzahl von 0 in } w\text{"}$

### Übungsaufgabe Ü3.6. (Ardens Lemma)

Gegeben sei folgender Automat  $M = (\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{q_3\})$ :



Utkan Sulejmani

Berechnen Sie mit dem Gauß-Verfahren und Ardens Lemma einen regulären Ausdruck  $\alpha$  mit  $L(\alpha) = L(M)$ .

### Übungsaufgabe Ü3.7. (Strukturelle Induktion: Klausuraufgabe von 2020)

- (a) Sei  $\Sigma$  ein Alphabet. Geben Sie die Rekursionsgleichungen für eine rekursive Prozedur contains( $a, r$ ), die für einen Buchstaben  $a \in \Sigma$  und regulären Ausdruck  $r$  berechnet, ob  $a$  in jedem Wort aus  $L(r)$  vorkommt.  
Es soll also für alle  $w \in L(r)$  gelten, dass  $(\exists u, v \in \Sigma^* . w = uav) =: P(w)$ .

- (b) Beweisen Sie mit Hilfe von struktureller Induktion, dass ihre Prozedur korrekt ist. Sie dürfen dabei den Konkatenationsfall (d.h.  $r = r_1r_2$ ) weglassen. Kennzeichnen Sie dabei die Induktionshypotesen und deren Anwendungen deutlich.

$$f : RE \longrightarrow \text{bool} \quad L(\underline{\alpha}) = \{\alpha\}$$

$$\begin{array}{lll} f \neq & = & \text{contains}(a, \underline{\alpha}) \\ f \underline{a} & = & = (\alpha = a) \\ f \underline{e} & = & \end{array}$$

$$f(r_1r_2) = \underbrace{f(r_1)}_{;} - \dots - \underbrace{f(r_2)}_{;}$$

## Strukt. Induktion

$$\text{nat} = 0 \mid S_{\text{nat}}$$

$$0 \in \mathbb{N}$$

$$n \in \mathbb{N} \Rightarrow \text{Succ}(n) \in \mathbb{N}$$

- $\emptyset \in \text{RegExp}$

- $a \in \text{RegExp}_{\geq 0}$  (für alle  $a \in \Sigma$ )

- $\underline{\underline{e}} \in \text{RegExp}$

- $r_1, r_2 \in \text{RE} \Rightarrow r_1 r_2 \in \text{RE}$

- $r_1, r_2 \in \text{RE} \Rightarrow r_1 / r_2 \in \text{RE}$

- $r \in \text{RE} \Rightarrow r^* \in \text{RE}$

$$L_2 = \{w \in \{0,1\}^* \mid |w|_0 \geq |w|_1\}$$

Angenommen,  $L_2$  wäre regulär.

Sei  $n$  eine PL-Zahl und  $z = 1^n 0^n \in L_2$ .

Es gilt  $|z| \geq n$ . Seien  $u, v, w \in \{0,1\}^*$ , sodass

$$1) \quad z = uvw$$

$$2) \quad |uv| \leq n$$

$$3) \quad v \neq \epsilon.$$

Daraus folgt,  $uv = 1^k$  für  $0 < k \leq n$  und somit  $v = 1^i$  mit  $0 < i \leq k$ .

$$\underbrace{1 \dots 1}_{i} \underbrace{00 \dots 0}_{n-i}$$

$$i = k$$

$$11 \dots 1 \ 00 \dots 0$$

$$\left|uv^2w\right|_1 = \begin{cases} = n & \\ = i > 0 & \end{cases} = i + |v|_1$$

$$i \leq k \leq n$$

$$> n$$

Somit ist  $uv^2w \notin L_2$ .

Das ist ein Widerspruch, also  $L_2$  nicht =

$$\left|uv^2w\right|_0 = 0 \\ \left|uvw\right|_0 + \left|vb\right|_0$$

$$L_1 = \{ w \in \{0,1\}^* \mid w = w^2 \}$$

$uv^2w \notin L$

$uv^0w \in L$

$000 \in L_1$

$\forall z \in L$ . gibt es eine Zerlegung

$$z = \textcolor{red}{uvw}$$

$$uv^k w \in L$$

Wir wollen für unser bestimmtes  $z$ , dass alle Zerlegungen "scheitern", d.h. wenn alle Zerleg. aufgezählt werden ( $uv^iw$ ), die nicht in  $L$  sind

$(a, \phi) = \text{true}$   
 $(a, e) = \text{false}$   
 $(a, z) = (x = a)$   
 $(a, r_1 r_2) = \text{concat}(a, r_1) \vee \text{concat}(a, r_2)$   
 $(a, r_1 | r_2) = \text{concat}(a, r_1) \wedge$   
 $(a, r^\omega) = \text{false}$

contains

zu zeigen

$\text{contains}(a, r)$   
 $\Leftrightarrow \exists w \in L(r). P(w)$   
 Eigenschaft  
vom Wort

z.B. Fall:  $r = \emptyset$ .

$\exists w \in L(\emptyset). P(w)$

$\Leftrightarrow \text{true}$   
 $\Leftrightarrow \text{contains}(a, \emptyset)$

Fall:  $r = \Sigma$

$\forall w \in L(\Sigma). P(w)$

$\Leftrightarrow \forall w \in \{\infty\}. P(w)$

$\Leftrightarrow P(\{\infty\})$

$\Leftrightarrow \underline{(x = a)}$

$\Leftrightarrow \text{contains}(a, \Sigma)$

$L(\Sigma) = \{x\}$

Fall:  $r = r_1 \mid r_2$

IH:  $\text{contains}(a, r_i) \iff \exists w \in L(r_i). P(w)$

Schritt:

$(\exists w \in L(r_1 \mid r_2). P(w))$

$\iff (\exists w \in L(r_1) \cup L(r_2). P(w))$

$\iff (\exists w \in L(r_1) . P(w))$

$\wedge (\exists w \in L(r_2) . P(w))$

(IH)

$\iff$

$\text{contains}(a, r_1)$

$\wedge \text{contains}(a, r_1)$

(Def.)

$\iff$

$\text{contains}(a, r_1 \mid r_2)$



$\forall x \in A \cup B. Q(x)$

$\iff$

$(\forall x \in A. Q(x))$

$\wedge (\forall x \in B. Q(x))$

Fall:  $r = r_1^*$ .

IH:  $\text{contains}(a, r_1) \Leftrightarrow \forall w \in L(r_1). P(w)$

Schritt: Es gilt  $e \in L(r_1^*) = (L(r_1))^*$ . Also  
gilt  $\text{contains}(a, r_1^*)$  nicht. Damit ist  
 $\text{contains}(a, r_1^*) = \text{false}$  richtig.

$$\underline{L^* = \{e\} \cup L^+}$$

$\forall w \in L(r_1^*) . P(w)$   
 $\Leftrightarrow \forall w \in L(r_1)^* . P(w)$   
 $\Leftrightarrow (\forall w \in \underline{L(r_1)^+} . P(w)) \wedge (\forall w \in \underline{\{e\}} . P(w))$   
 $\Leftrightarrow \text{false}$   
 $\Leftrightarrow \text{contains}(a, r_1^*)$

$$L_1 = \{ w \in \{0,1\}^* \mid w = w^r \}$$

Nehne an:  $L_1$  regulär.

Sei  $n$  eine PL-Zahl und sei  $z := 0^n 1 0^n \in L$ . Es gilt  $|z| \geq n$ . Seien  $u, v, w$  beliebig s.d.  $z = uvw$  und s.d.  $|uv| \leq n$  und  $v \neq \epsilon$ . Es muss ja gelten:

$$uv = 0^i$$

und  $w = 0^{n-i} 1 0^n$

für  $i \leq n$ .

Da  $v \neq \epsilon$ , gilt  $v = 0^k$  für  $0 < k \leq i$ . Also gilt nach PL

$$uv^0 w = 0^{i-k} 1 0^n \in L.$$

Aber: das ist ein Widerspruch, denn

$$\begin{aligned} 0^{i-k} 1 0^n &\neq (0^{i-k} 1 0^n)^r \\ &= 0^n 1 0^{i-k} \end{aligned}$$

wegen  $i-k \neq n$ .

$n=3$

$|uv| \leq 3$

~~000 1000~~

$\rightarrow$  ~~01000~~  $\notin L$ .