Prof. Dr. Jasmin Blanchette
Elisabeth Lempa
Luca Maio

Ludwig-Maximilians-Universität München
Institut für Informatik
Besprechung 26.06.2025 bis 30.06.2025
Abgabe bis 07.07.2025, 10:00 Uhr

Lösungsvorschlag zur Übung 8 zur Vorlesung
Formale Sprachen und Komplexität

FSK8-1 ***Entscheiden des Wortproblems für Sprachen von Typ 1***

Sei $G = (V, \{a, b, c\}, P, S)$ eine Grammatik mit Produktionen

$$\begin{aligned} P = \{S &\to SabS \mid Ac \mid cB, \\ Ac &\to ccA, \\ cA &\to ca, \\ AA &\to aA, \\ Aa &\to ab, \\ cB &\to Bcc, \\ Bc &\to bc, \\ BB &\to Bb, \\ bB &\to ab\} \end{aligned}$$

a) Berechnen Sie gemäß der Konstruktion aus der Vorlesung L_i^6 für alle $i \in \mathbb{N}$ und begründen Sie kurz.

b) Geben Sie zwei Wörter an, die auf Grundlage der Berechnung in der Sprache $L(G)$ sind.

c) Geben Sie drei Wörter an, die auf Grundlage der Berechnung nicht in der Sprache $L(G)$ sind.

LÖSUNGSVORSCHLAG:

a)

$L_0^6 := \{S\}$

$L_1^6 := L_0^6 \cup \{SabS, Ac, cB\}$,
da $S \to SabS \mid Ac \mid cB$

$L_2^6 := L_1^6 \cup \{AcabS, cBabS, SabAc, SabcB, ccA, Bcc\}$,
da $S \to Ac \mid cB$, $Ac \to ccA$, $cB \to Bcc$

$L_3^6 := L_2^6 \cup \{AcabAc, AcabcB, ccAabS, cBabAc, SabccA, cBabcB, BccabS, SabBcc, cca, bcc\}$,
da $S \to Ac \mid cB$, $Ac \to ccA$, $cA \to ca$, $Bc \to bc$

$L_4^6 := L_3^6 \cup \{ccaabS, ccabbS, Sabcca, bccabS, Sabbcc, Saabcc\}$,
da $Aa \to ab$, $bB \to ab$, $cA \to ca$, $Bc \to bc$

$L_i^6 := L_4^6$

für $i \geq 4$, da alle anwendbaren Regeln zu Satzformen mit Länge größer 6 führen.

b) cca, bcc

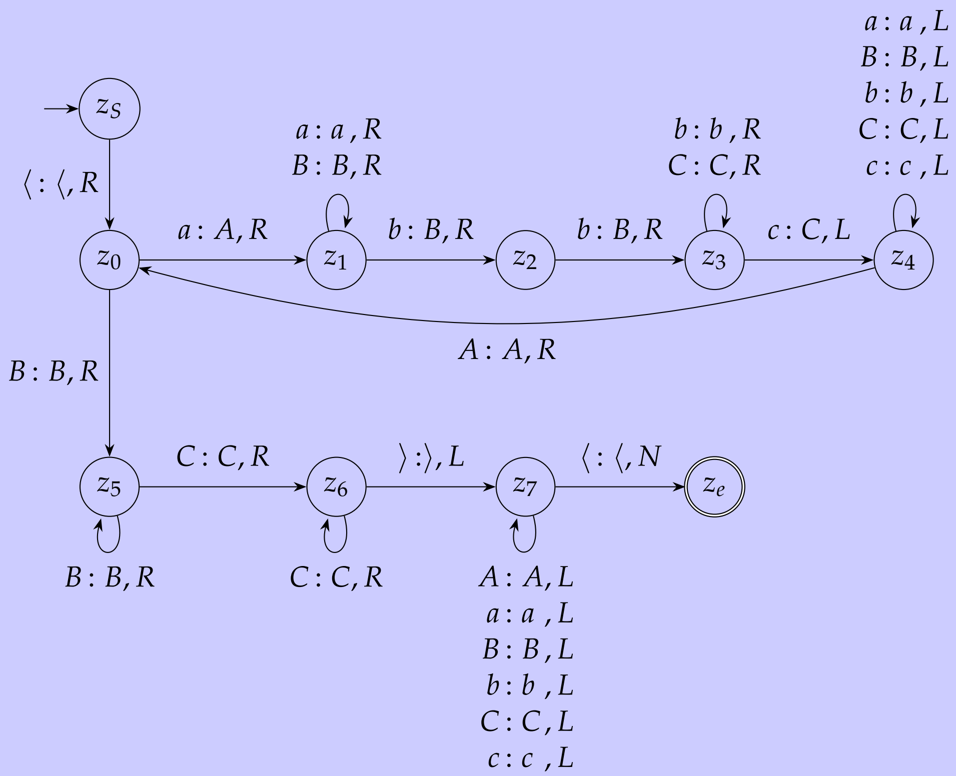
c) a, b, c

FSK8-2 ***LBAs***

Wir betrachten die Sprache $L = \{a^n b^{2n} c^n \mid n \in \mathbb{N}_{>0}\}$ über dem Alphabet $\{a, b, c\}$.

a) Geben Sie einen Zustandsgraphen für einen LBA an, der L erkennt.

LÖSUNGSVORSCHLAG:
LBA: `http://turingmachinesimulator.com/shared/odgepbeprk`
Hierbei müssen Sie manuell $<$ anstelle $\langle$ und $>$ anstelle $\rangle$ um das Eingabewort schreiben.



b) Geben Sie eine Definition dafür an, dass ein LBA "deterministisch" ist in dem Sinne wie wir es für die anderen Konstrukte dieser Vorlesung getan haben.

LÖSUNGSVORSCHLAG:
Ein LBA $M = (Z, \Sigma, \Gamma, \delta, z_0, \langle, \rangle, E)$ ist deterministisch g.d.w. $|\delta(z, a)| = 1$ für alle $z \in Z, a \in \Gamma$.

c) Geben Sie eine Definition dafür an, dass ein nach Ihrer Definition aus b) deterministischer LBA eine Funktion berechnet. Orientieren Sie sich dabei an der entsprechenden Definition für deterministische Turingmaschinen.

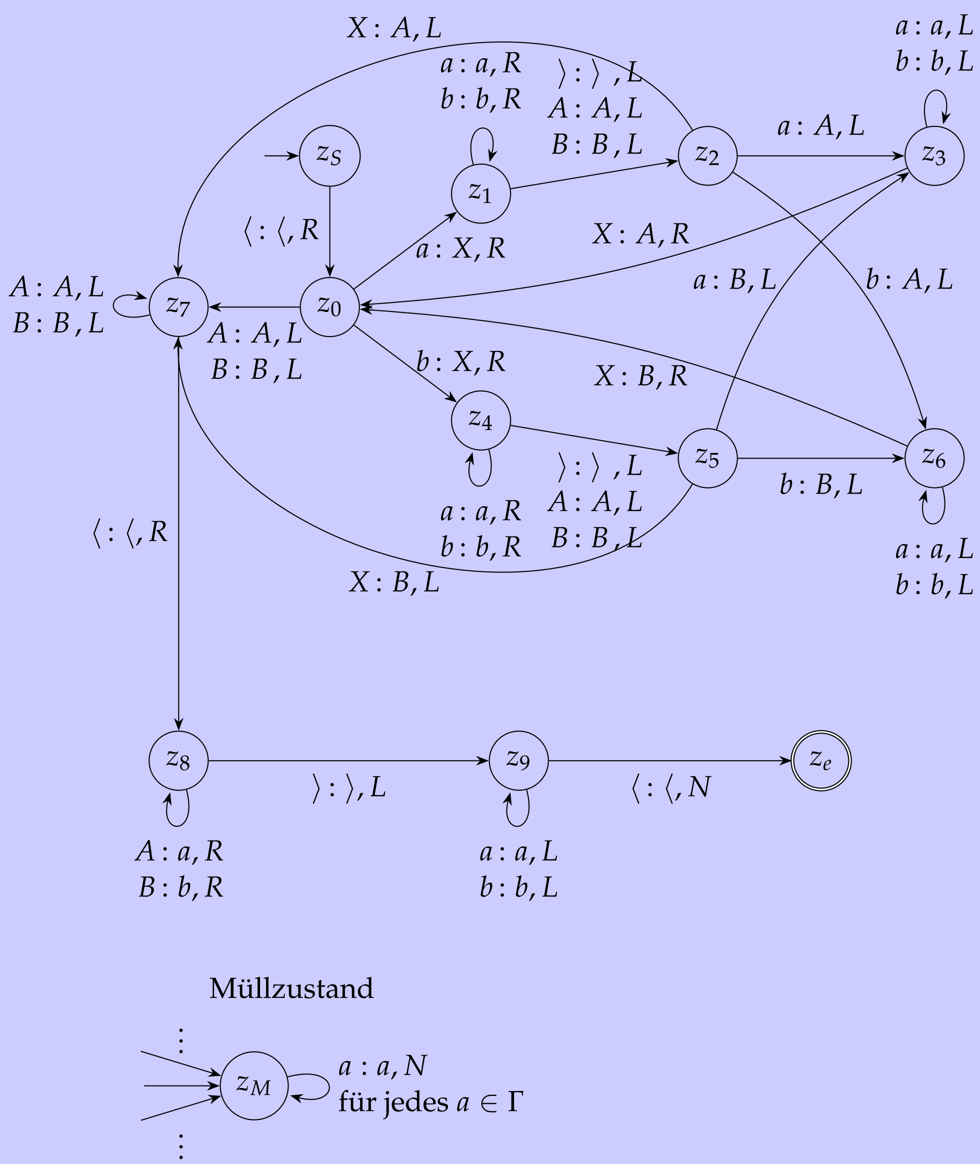
LÖSUNGSVORSCHLAG:
Ein LBA $M = (Z, \Sigma, \Gamma, \delta, z_0, \langle, \rangle, E)$ berechnet eine Funktion $f : \Sigma^* \to \Sigma^*$, wenn für alle $u, v \in \Sigma^*$ gilt: $f(u) = v$ g.d.w.

$$z_0\langle u \rangle \vdash^* \Box \cdots \Box z_e \langle v \rangle \Box \cdots \Box \text{ mit } z_e \in E.$$

d) Geben Sie den Zustandsgraphen eines nach Ihrer Definition aus b) deterministischen LBA an, der nach Ihrer Definition aus c) die Funktion $f(w) = \overline{w}$ über dem

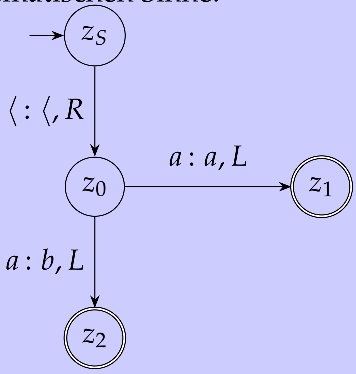
Alphabet $\Sigma = \{a, b\}$ berechnet.

LÖSUNGSVORSCHLAG:



e) Warum ist es für die Definition aus c) notwendig, dass der LBA deterministisch ist?

LÖSUNGSVORSCHLAG:
Ein nichtdeterministischer LBA könnte eine "Funktion" definieren, welche für denselben Input mehrere Outputs hat. Dies ist dann aber keine Funktion im mathematischen Sinne.

Beispiel: 

FSK8-3 ***Turingmaschinen***

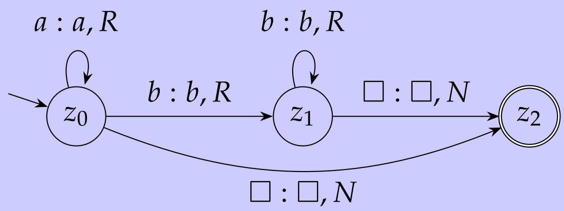
a) Seien $M = \{w \mid \#_a(w) = \#_b(w)\}$ und $N = L(a^*b^*)$ Sprachen über $\Sigma = \{a, b\}$. Zeigen Sie, dass es eine Funktion f gibt mit $\forall x \in \Sigma^* . x \in M \iff f(x) \in N$.

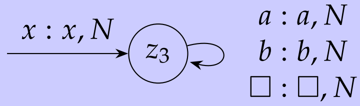
LÖSUNGSVORSCHLAG:

$$f(x) = \begin{cases} ab & \text{wenn } \#_a(x) = \#_b(x) \\ ba & \text{sonst} \end{cases}$$

b) Geben Sie eine deterministische Turingmaschine an, welche N erkennt.

LÖSUNGSVORSCHLAG:



Fehlende Übergänge mit $x \in \{a, b, \square\}$: 

c) Was muss für f gelten, damit man daraus folgern kann, dass M von einer Turingmaschine erkannt werden kann (ohne die Turingberechenbarkeit von M direkt zu zeigen)?

LÖSUNGSVORSCHLAG: f muss turingberechenbar sein. Sei F die passende Turingmaschine. Dann kann man M erkennen, indem man zunächst F ausführt und danach die Turingmaschine für N ausführt. Wie wir im Laufe der Vorlesung sehen werden, nennt man f in diesem Fall eine Reduktionsfunktion, für eine Reduktion von M auf N.