

Lösungsvorschlag zur Übung 7 zur Vorlesung
Formale Sprachen und Komplexität

FSK7-1 Sprachen einordnen

Die formalen Sprachen $L_i, i \in \{1, \dots, 3\}$, seien definiert als

$$\begin{aligned} L_1 &:= \{ab^i \mid i \in \mathbb{N}\} \cup \{c^i a \mid i \in \mathbb{N}\} && \subseteq \{a, b, c\}^* \\ L_2 &:= \{(ab)^j \$ c^i \mid i, j \in \mathbb{N}\} && \subseteq \{a, b, c, \$\}^* \\ L_3 &:= \{(ab)^i \$ c^j \$ (ab)^i \mid j < i \text{ und } i, j \in \mathbb{N}\} && \subseteq \{a, b, c, \$\}^* \end{aligned}$$

Für die i -fache Wiederholung des Worts w schreiben wir manchmal $(w)^i$ statt nur w^i , um Anfang und Ende von w zu markieren. Die Klammern sind daher *nicht* Teil des Alphabets der jeweiligen Sprachen.

Bearbeiten Sie die folgenden Arbeitsaufträge für jede der Sprachen L_i .

- Beweisen oder widerlegen Sie, dass L_i regulär ist.
- Beweisen oder widerlegen Sie, dass L_i kontextfrei ist.

Hinweis: Nutzen Sie, dass manche Aussagen direkt aus anderen Aussagen folgen. Um zu beweisen, dass L_i regulär/kontextfrei ist, genügt es, ein geeignetes Konstrukt K_i (Grammatik, Automat oder regulärer Ausdruck) anzugeben und kurz zu begründen, warum $L(K_i) = L_i$ gilt.

LÖSUNGSVORSCHLAG:

- $L_1 = \{ab^i \mid i \in \mathbb{N}\} \cup \{c^i a \mid i \in \mathbb{N}\} = L(ab^*) \cup L(c^*a)$ ist regulär (und damit auch kontextfrei), da die Sprache von einer Vereinigung regulärer Ausdrücke erzeugt wird. Jedes Wort der Sprache ist entweder aus ab^* oder aus c^*a , da beide i 's unabhängig voneinander sind. Wir nutzen hierbei, dass reguläre Sprachen unter Vereinigung abgeschlossen sind.
- $L_2 = \{(ab)^j \$ c^i \mid i, j \in \mathbb{N}\} = L((ab)^* \$ c^*)$ ist regulär (und damit auch kontextfrei), da die Sprache von einem regulären Ausdruck erzeugt wird. Jedes Wort der Sprache besteht erst aus beliebig vielen ab 's gefolgt von $\$$ und endet mit beliebig vielen c .
- L_3 ist nicht kontextfrei und damit auch nicht regulär. Wir widerlegen die

Kontextfreiheit mit dem Pumping-Lemma für kontextfreie Sprachen.

Der Beweis ist durch Widerspruch. Wir nehmen an, L_3 ist kontextfrei, d.h. die Pumping-Eigenschaft gilt für L_3 .

Gegeben sei $n \in \mathbb{N}_{>0}$. Wir wählen $z = (ab)^{2n}c^n(ab)^{2n}$. Dann gilt $|z| \geq n$ und $z \in L_3$.

Sei $z = uvwxy$ eine beliebige Zerlegung von z mit $|vwx| \leq n$, $|vx| > 0$ und $uv^iwx^iy \in L_3$ für alle $i \in \mathbb{N}$.

Wenn vx das Symbol c enthält, dann gilt $uv^2wx^2y \notin L_3$, da $\#_c(uv^2wx^2y) > 2$. Sonst enthält vx entweder a oder b aus genau einem der beiden $(ab)^{2n}$ -Blöcke.

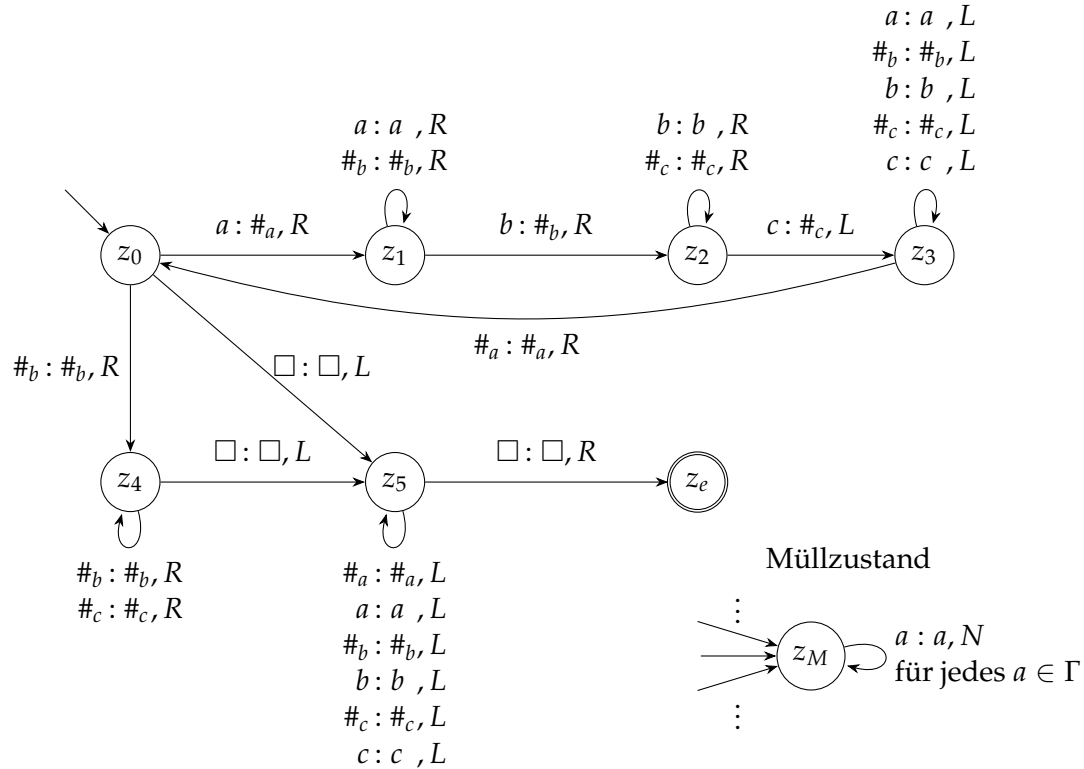
Enthält vx nun c , so wählen wir $i = 2n$ und damit ist $\#_c(uv^iwx^iy) \geq 3n - 1 \geq 2n$ aber vx enthält a und b aus höchstens einem $(ab)^{2n}$ -Block (da $|vwx| \leq n$) und somit ist mindestens einer der beiden $(ab)^{2n}$ -Blöcke zu klein um $j < i$ zu erfüllen. Somit gilt $uv^iwx^iy \notin L_3$. Dies verletzt aber die Annahme, dass $uv^iwx^iy \in L_3$ für jedes i . Widerspruch.

Enthält vx kein c , dann muss vx aufgrund von $|vx| > 0$ mindestens ein a oder b aus genau einem der $(ab)^{2n}$ -Blöcke enthalten, da $|vwx| \leq n$. Dann wählen wir $i = 0$, womit in uv^iwx^iy genau einer der $(ab)^{2n}$ -Blöcke weniger als $4n$ Symbole enthält und damit gilt $uv^iwx^iy \notin L_3$. Dies verletzt aber die Annahme, dass $uv^iwx^iy \in L_3$ für jedes i . Widerspruch.

L_3 ist somit nicht kontextfrei (und damit auch nicht regulär).

FSK7-2 Turingmaschinen verstehen

Die folgende DTM M ist als Zustandsgraph gegeben, wobei $\Sigma = \{a, b, c\}$, $\Gamma = \Sigma \cup \{\#_a, \#_b, \#_c, \square\}$ und $\square$ das Blank-Symbol ist.



LÖSUNGSVORSCHLAG:

Simulator: <http://turingmachinesimulator.com/shared/pvchzxkffi>

- a) Geben Sie Läufe der Turingmaschine (Übergänge von der Startkonfiguration bis zur Endkonfiguration) für die Wörter ε , $abcc$ und abc an.

LÖSUNGSVORSCHLAG:

- $z_0 \square \vdash z_5 \square \vdash z_e \square$
- $z_0 abcc \vdash \#_a z_1 bcc \vdash \#_a \#_b z_2 cc \vdash \#_a z_3 \#_b \#_c c \vdash z_3 \#_a \#_b \#_c c$
 $\vdash \#_a z_0 \#_b \#_c c \vdash \#_a \#_b z_4 \#_c c \vdash \#_a \#_b \#_c z_4 c \vdash \#_a \#_b \#_c z_M c \vdash \dots$
- $z_0 abc \vdash \#_a z_1 bc \vdash \#_a \#_b z_2 c \vdash \#_a z_3 \#_b \#_c \vdash z_3 \#_a \#_b \#_c$
 $\vdash \#_a z_0 \#_b \#_c \vdash \#_a \#_b z_4 \#_c \vdash \#_a \#_b \#_c z_4 \square \vdash \#_a \#_b z_5 \#_c$
 $\vdash \#_a z_5 \#_b \#_c \vdash z_5 \#_a \#_b \#_c \vdash z_5 \square \#_a \#_b \#_c \vdash z_e \#_a \#_b \#_c$

- b) Welche Sprache akzeptiert die Turingmaschine M ? Begründen Sie Ihre Antwort.

LÖSUNGSVORSCHLAG:

Die Turingmaschine akzeptiert $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. Dazu geht sie wie folgt vor:

- In den Zuständen z_0 , z_1 und z_2 wird ein a , dann ein b und dann ein c gesucht, wobei jeweils ein gefundener Buchstabe x durch $\#_x$ ersetzt wird. Dabei überspringt die TM in z_1 nur a 's und $\#_b$'s und in z_2 nur b 's und $\#_c$'s, das heißt es müssen tatsächlich zuerst die a 's, dann die b 's und dann die c 's im Wort auftreten.
- Im Zustand z_3 fährt die TM zurück zum ersten a , das noch nicht verarbeitet wurde. Somit werden in einer Schleife alle a 's mit b 's und c 's gepaart.
- Wenn die TM in z_0 ein $\#_b$ liest, sind alle a 's abgearbeitet. Wir müssen dann nur noch sicherstellen, dass keine b 's oder c 's mehr im Wort sind, und laufen deswegen mit z_4 noch einmal über $\#_b$'s und $\#_c$'s bis an das Wortende.
- Wenn das gelingt, fahren wir mit z_5 wieder an den Anfang des Wortes und akzeptieren.
- Spezialfall: Wenn wir in z_0 ein $\square$ lesen, ist das Wort leer und wir akzeptieren via z_4 und z_5 sofort.

- c) Die obige Turingmaschine M mit Alphabet Σ und Bandalphabet Γ berechnet eine (partielle) Funktion $f : \Sigma^* \rightarrow \Gamma^*$, wenn für alle $u \in \Sigma^*$ und $v \in \Gamma^*$ gilt: $f(u) = v$ g.d.w. $z_0 u \vdash_M^* \square \cdots \square z_e v \square \cdots \square$ mit z_e Endzustand. (Beachten Sie: Diese Definition weicht leicht von der aus der Vorlesung ab, weil die Wertemenge von f nicht Σ^* ist, sondern Γ^* .)

Welche Funktion berechnet M ?

LÖSUNGSVORSCHLAG:

Wie in Teilaufgabe b) gezeigt, erkennt M genau die Wörter der Form $a^n b^n c^n$. Im Endzustand sind dabei alle a 's durch $\#_a$'s ersetzt und analog für b und c . Somit berechnet M folgende Funktion f :

$$f(w) = \begin{cases} \#_a^n \#_b^n \#_c^n & \text{für } w = a^n b^n c^n \\ \text{undefiniert} & \text{andernfalls} \end{cases}$$

- d) Bestimmen Sie asymptotisch, also in O -Notation, die Anzahl der Schritte (abhängig von n), die die Turingmaschine braucht, um das Wort $w = a^n b^n c^n$ zu

erkennen.

LÖSUNGSVORSCHLAG:

Wir zählen die Schritte eines erfolgreichen Laufs auf w .

- Von z_0 zu z_1 kommen wir in einem Schritt, der asymptotisch vernachlässigbar ist. Von z_1 zu z_2 und z_2 zu z_3 kommen wir in jeweils n Schritten (da wir immer das i -te a mit dem i -ten b und i -ten c paaren), also insgesamt in $2n$ oder $O(n)$ Schritten.
- Von z_3 zu z_0 kommen wir in mindestens $2n$ und höchstens $3n$, also $O(n)$ Schritten.
- Die Schleife von z_0 bis z_3 wird n -mal durchlaufen. Da jeder Durchlauf $O(n)$ Schritte benötigt, benötigt die gesamte Schleife $O(n^2)$ Schritte.
- Via z_4 laufen wir noch einmal in $2n$, also $O(n)$, Schritten über das ganze Wort.
- Am Ende laufen wir noch in $3n$, also $O(n)$, Schritten zurück auf die Startposition.

Die gesamte Laufzeit ist also $O(n^2)$.

FSK7-3 Turingmaschinen erstellen

Wir betrachten die Sprache $L = \{w \mid \#_b(w) > \#_c(w)\}$ über dem Alphabet $\{b, c\}$.

- a) Erstellen Sie auf <https://turingmachinesimulator.com/> eine TM, die L erkennt. Geben Sie sowohl einen Link zur Maschine an als auch den „Programmtext“ der Maschine.

LÖSUNGSVORSCHLAG:

TM: <http://turingmachinesimulator.com/shared/ndqrvoyaon>

Idee: Wir gehen von links nach rechts über das Wort. Für jedes b , das wir dabei finden, suchen wir nach rechts ein entsprechendes c , und für jedes c suchen wir ein b . Die von links abgearbeiteten b 's und c 's ersetzen wir durch X und markieren so den Teil des Wortes, den wir nicht mehr besuchen müssen. Die entsprechenden c 's und b 's ersetzen wir durch C 's und B 's, um zu markieren, dass sie bereits verwendet wurden. Wenn wir nach einem „Partner“ zu einem b suchen, aber keinen finden, also $\square$ lesen, akzeptieren wir das Wort, da es somit mindestens ein mehr b 's als c 's geben muss.

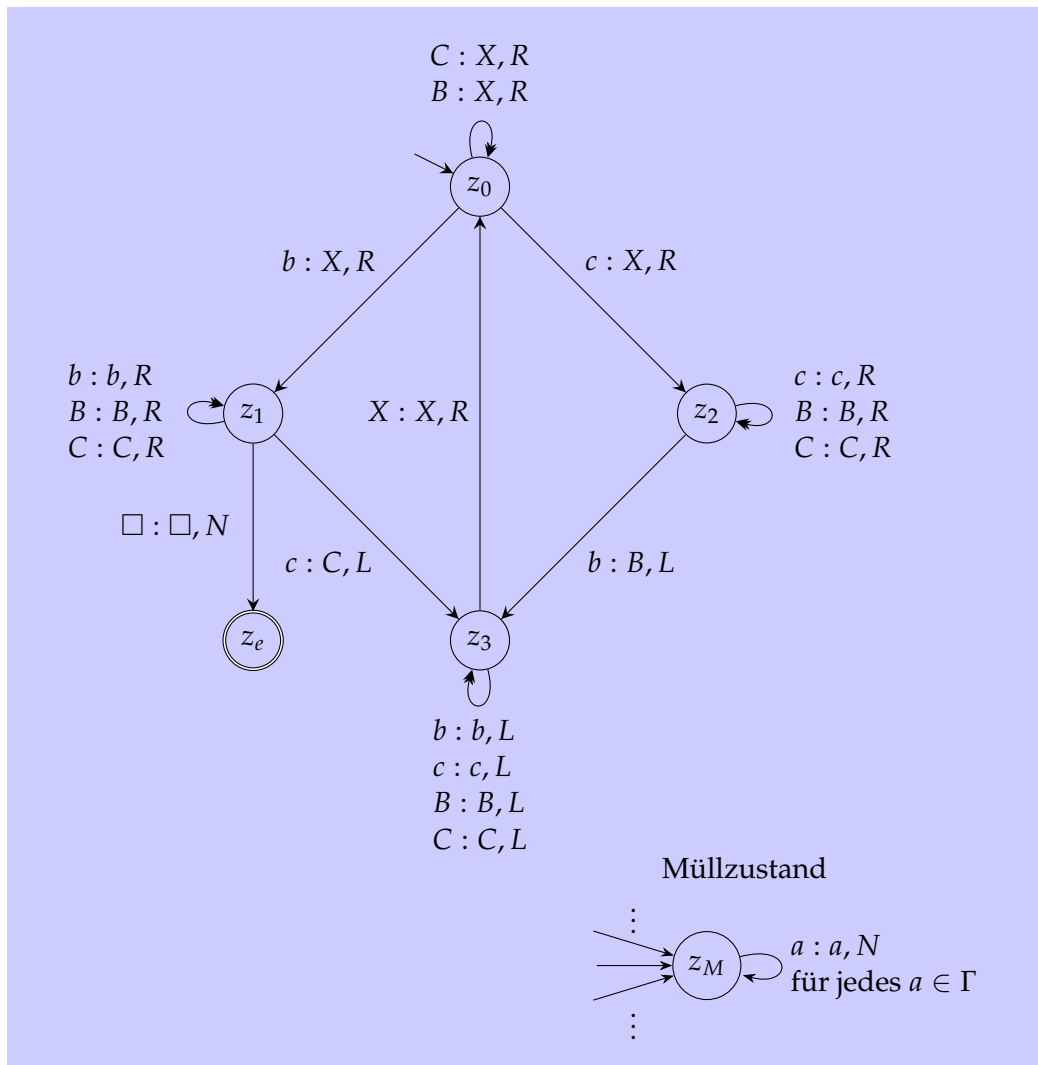
Genauer führen wir folgende Schritte in einer Schleife aus:

- i) (z_0 , Startzustand) Laufe nach rechts über das Wort und halte nach dem ersten b oder c an. Das gefundene b oder c wird dabei durch X ersetzt. Ist das erste Zeichen bereits ein b oder c , halte auf dem zweiten Zeichen.
- ii) (z_1) Ist das gefundene Zeichen ein b , laufe nach rechts und ersetze das erste c durch ein C .
- iii) (z_2) Ist das gefundene Zeichen ein c , suche entsprechend nach dem ersten b und ersetze es durch ein B .
- iv) (z_3) Laufe nach links über das Wort und halte rechts des ersten X .

Wir akzeptieren, wenn wir in $z_1 \square$ lesen, es also kein c zum bereits gelesenen b gibt. Als Optimierung ersetzen wir im ersten Schritt B und C durch X und verkürzen so den Teil des Wortes, den wir in den nächsten Durchläufen betrachten müssen. Das ist aber nicht nötig.

- b) Geben Sie für Ihre TM aus Teilaufgabe a) einen Zustandsgraphen an.

LÖSUNGSVORSCHLAG:



- c) Geben Sie die Läufe der folgenden Wörter auf Ihre TM aus Teilaufgabe a) an: $\varepsilon, c, bcc, cbcb$.

Hinweis: Wörter, die nicht in L liegen, erzeugen eventuell unendliche Läufe. Geben Sie in solchen Fällen ein Präfix an, aus dem ersichtlich wird, dass der Lauf unendlich ist.

LÖSUNGSVORSCHLAG:

Für ε :

$$z_0 \square \vdash z_M \square \vdash z_M \square \vdash \dots$$

Für c :

$$z_0 c \vdash X z_2 \square \vdash X z_M \square \vdash X z_M \square \vdash \dots$$

Für bcc :

$$z_0bcc \vdash Xz_1cc \vdash z_3XCc \vdash Xz_0Cc \vdash XXz_0c \vdash XXXz_2\Box \vdash XXXz_M\Box \\ \vdash XXXz_M\Box \vdash \dots$$

Für $cbcb$:

$$z_0cbcb \vdash Xz_2cbcb \vdash z_3XBcb \vdash Xz_0Bcb \vdash XXz_0cb \vdash XXXz_2bb \\ \vdash XXz_3XBb \vdash XXXz_0Bb \vdash XXXXz_0b \vdash XXXXXz_1\Box \\ \vdash XXXXXz_e\Box$$

FSK7-4 Konstruktion Grammatik zu PDA

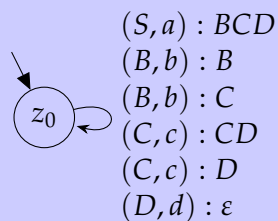
Sei $G = (\{S, B, C, D\}, \{a, b, c, d\}, P, S)$ eine Grammatik in Greibach-Normalform mit Produktionen

$$P = \{S \rightarrow aBCD, B \rightarrow bB \mid bC, C \rightarrow cCD \mid cD, D \rightarrow d\}$$

- a) Erzeugen Sie gemäß der Konstruktion aus der Vorlesung aus G einen PDA M mit $L(M) = L(G)$, der mit leerem Keller akzeptiert.

LÖSUNGSVORSCHLAG:

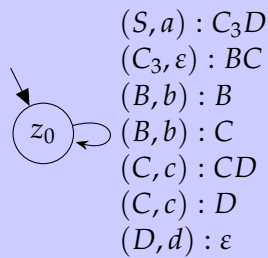
Zustandsgraph zu PDA M :



- b) Erzeugen Sie gemäß der sogenannten Tripelkonstruktion aus der Vorlesung aus M eine kontextfreie Grammatik H mit $L(H) = L(M)$.

LÖSUNGSVORSCHLAG:

Zuerst formen wir M um sodass für alle Übergänge gilt, dass wir maximal 2 Symbole auf den Stack legen, hierfür verwenden wir die Konstruktion aus Lemma 5.7.12 im Skript:



Sei $H = (V, \{a, b\}, P, S')$ mit Variablen

$$V = \{S', \langle z_0, S, z_0 \rangle, \langle z_0, B, z_0 \rangle, \langle z_0, C, z_0 \rangle, \langle z_0, D, z_0 \rangle, \langle z_0, C_3, z_0 \rangle\}$$

und Produktionen

$$\begin{aligned}
 P = \{ & S' \rightarrow \langle z_0, S, z_0 \rangle, \\
 & \langle z_0, S, z_0 \rangle \rightarrow a \langle z_0, C_3, z_0 \rangle \langle z_0, D, z_0 \rangle, \\
 & \langle z_0, C_3, z_0 \rangle \rightarrow \langle z_0, B, z_0 \rangle \langle z_0, C, z_0 \rangle, \\
 & \langle z_0, B, z_0 \rangle \rightarrow b \langle z_0, B, z_0 \rangle, \\
 & \langle z_0, B, z_0 \rangle \rightarrow b \langle z_0, C, z_0 \rangle, \\
 & \langle z_0, C, z_0 \rangle \rightarrow c \langle z_0, C, z_0 \rangle \langle z_0, D, z_0 \rangle, \\
 & \langle z_0, C, z_0 \rangle \rightarrow c \langle z_0, D, z_0 \rangle, \\
 & \langle z_0, D, z_0 \rangle \rightarrow d \}
 \end{aligned}$$

- c) Vergleichen Sie die Grammatiken G und H . Beschreiben Sie die Gemeinsamkeiten dieser Grammatiken, sowie ihre Unterschiede.

LÖSUNGSVORSCHLAG:

Gemeinsamkeiten:

- Sie akzeptieren die gleiche Sprache (aufgrund der Konstruktion).
- Die Variablen aus G kommen in H prinzipiell wieder vor innerhalb der Tripel.
- Modulo Umbenennung der Variablen sind die Produktionen aus G alle in denen von H enthalten

Unterschiede:

- H enthält die zusätzliche Variablen S' und C_3 und allgemein haben alle Variablen die Tripelform. Die Tripel sind, da M nur einen Zustand hat, immer von der Form $\langle z_0, A, z_0 \rangle$.
- Aufgrund der neuen Produktionen $S' \rightarrow \langle z_0, S, z_0 \rangle$ und $\langle z_0, C_3, z_0 \rangle \rightarrow \langle z_0, B, z_0 \rangle \langle z_0, C, z_0 \rangle$ ist H nicht mehr in Greibach-Normalform.