Institut für Informatik der Ludwig-Maximilians-Universität München
Prof. Dr. Jasmin Blanchette
Lydia Kondylidou
Jannis Limperg

SoSe 2023
02.08.2023

Klausur zur Vorlesung
Formale Sprachen und Komplexität

Die Bearbeitungszeit beträgt **120 Minuten**. Hilfsmittel sind nicht erlaubt, auch das Mitführen ausgeschalteter elektronischer Geräte wird als Betrug gewertet. Schreiben Sie Ihren vollständigen Namen und Ihre Matrikelnummer deutlich lesbar auf dieses Deckblatt, sowie Ihren Namen in die Kopfzeile auf **jedem Blatt** der Klausurangabe. Geben Sie alle Blätter ab, lassen Sie diese zusammengeheftet. Verwenden Sie nur **dokumentenechte Stifte** und **weder** die Farben **rot noch grün**.

Kontrollieren Sie, ob Sie alle Aufgabenblätter erhalten haben. Aufgabenstellungen befinden sich auf den **Seiten 1–16**. Sie dürfen die Rückseiten für Nebenrechnungen nutzen. Falls Sie die Rückseiten für Antworten nutzen, so markieren Sie klar, was zu welcher Aufgabe gehört und geben Sie in der entsprechenden Aufgabe an, wo alle Teile Ihrer Antwort zu finden sind. Streichen Sie alles durch, was nicht korrigiert werden soll.

Lesen Sie die Aufgabenstellungen vollständig durch, bevor Sie mit der Bearbeitung der Aufgaben beginnen. Es gibt 5 unterschiedlich gewichtete Aufgaben zu insgesamt 100 Punkten. Mit 50 Punkten haben Sie sicherlich bestanden. Die Teilaufgaben können unabhängig voneinander bearbeitet werden.

Mit Ihrer Unterschrift bestätigen Sie, dass Sie zu Beginn der Klausur in ausreichend guter gesundheitlicher Verfassung sind und diese Klausurprüfung verbindlich annehmen.

Nachname:

Vorname:

Matrikelnummer:

Studiengang:

Die Klausur soll gewertet werden: ☐ **JA** ☐ **NEIN**

Hinweis: Ist keines der beiden Felder angekreuzt, so wird **JA** angenommen.

Hiermit erkläre ich die Richtigkeit der obigen Angaben: ______________________

(Unterschrift)

Die folgende Tabelle **NICHT** ausfüllen:

Aufgabe	1	2	3	4	5	$\sum$	Bonus	Note
Punkte	33	28	11	16	12	**100**		
Erreicht								

Aufgabe 1 (Reguläre Sprachen): (33 Punkte)

a) Die Sprache $L_1 \subseteq \{a,b\}^*$ sei definiert als

$$L_1 := \{uabv \mid u,v \in \{a,b\}^*\} \cup \{ubav \mid u,v \in \{a,b\}^*\}$$

L_1 enthält daher genau alle Wörter über $\{a,b\}$, die ab oder ba als Teilwort haben.

Geben Sie den Zustandsgraphen eines nichtdeterministischen endlichen Automaten (ohne ε-Übergänge) A_1 an, für den gilt: $L(A_1) = L_1$.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 1:

b) Die Sprache $L_2 \subseteq \{a, b, c\}^*$ sei definiert als

$$L_2 := \{w \in \{a, b, c\}^* \mid \#_c(w) = 3\}$$

Die Sprache L_2 enthält daher alle Wörter über $\{a, b, c\}$, die genau drei c's enthalten.

Geben Sie einen regulären Ausdruck α mit $L(\alpha) = L_2$ an.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 1:

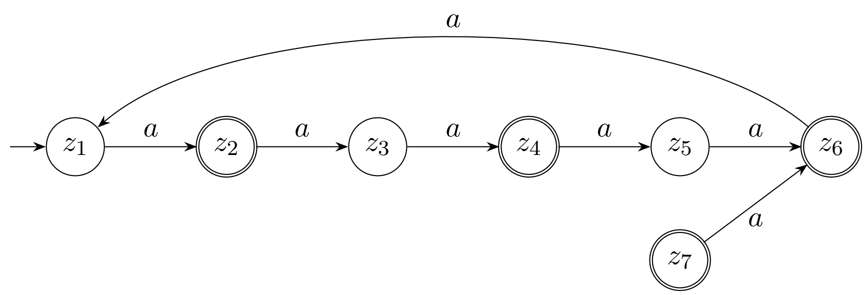
c) Bestimmen Sie den Index der Nerode-Relation für die Sprache $L_2 = \{w \in \{a,b,c\}^* \mid \#_c(w) = 3\}$ und begründen Sie Ihre Antwort.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 1:

d) Berechnen Sie für den folgenden deterministischen endlichen Automaten A_2 (der Wörter über dem Alphabet $\Sigma = \{a\}$ verarbeitet) den **Minimalautomaten**, d.h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt.

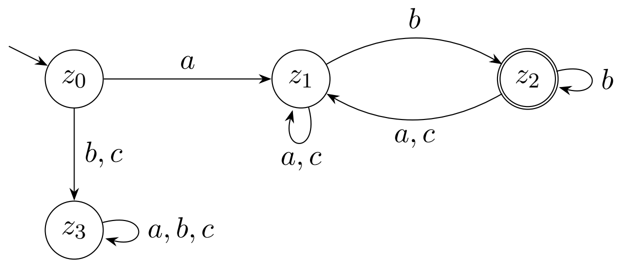
Erläutern Sie Ihre Berechnung, indem Sie die Minimierungstabelle angeben, und geben Sie den Minimalautomaten als Zustandsgraphen an.



Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 1:

e) Der folgende Automat A_3



ist bereits ein Minimalautomat mit Eingabealphabet $\Sigma = \{a, b, c\}$.

Begründen Sie für jedes der Zustandspaare

(i) (z_0, z_1)

(ii) (z_0, z_2)

(iii) (z_1, z_3)

warum die beiden Zustände des Paares **nicht äquivalent** sind.

Aufgabe 2 (Kontextfreiheit): (28 Punkte)

a) Die Sprache L_3 über $\{a, b, c, d\}$ sei definiert als

$$L_3 := \{a^i b^i c^j d^j \mid i, j \in \mathbb{N}_{>0}\}$$

Informell: Alle Wörter aus L_3 sind von der Form $a^i b^i c^j d^j$, wobei i, j positive Zahlen sind.

(i) Geben Sie eine **kontextfreie Grammatik** G_1 an, die L_3 erzeugt (d.h. $L(G_1) = L_3$) und erläutern Sie, warum G_1 die Sprache L_3 erzeugt. Beschreiben Sie beispielsweise, welche „Aufgabe“ die einzelnen Nichtterminale bei der Erzeugung übernehmen.

Geben Sie zusätzlich eine **Linksableitung** für das Wort *aabbcccddd* für Ihre Grammatik an.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 2:

(ii) Zeigen Sie mit dem Pumping-Lemma für reguläre Sprachen, dass

$$L_3 = \{a^i b^i c^j d^j \mid i, j \in \mathbb{N}_{>0}\}$$

nicht regulär ist.

Zur Erinnerung:

Pumping-Lemma für reguläre Sprachen.
Sei L eine reguläre Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$, das Mindestlänge n hat (d.h. $|z| \geq n$), als $z = uvw$ geschrieben werden kann, sodass gilt:
i) $|uv| \leq n$,
ii) $|v| \geq 1$ und
iii) für alle $i \geq 0$: $uv^i w \in L$.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 2:

b) Für eine kontextfreie Grammatik G_2 mit Startsymbol S und Variablen $\{S, A, B\}$ wurde der CYK-Algorithmus für das Wort *baabab* ausgeführt. Der Algorithmus hat dabei die folgende Tabelle für die $V(i, j)$-Mengen berechnet:

Wort	b	a	a	b	a	b
$j \backslash i$	1	2	3	4	5	6
1	B	A	A	B	A	B
2		B	S		S	
3			S	S		
4			A			
5		B, S				
6	S					

(i) Geben Sie anhand obiger Tabelle drei verschiedene Wörter an, die in $L(G_2)$ liegen und begründen Sie Ihre Antwort.

(ii) Geben Sie anhand obiger Tabelle zwei verschiedene Wörter an, die **nicht** in $L(G_2)$ liegen und begründen Sie Ihre Antwort.

Aufgabe 3 (Kontextsensitivität): (11 Punkte)

Sei $G_3 = (V = \{S, X\}, \Sigma = \{a, b, c\}, P, S)$ eine Typ 1-Grammatik, wobei

$$P = \{S \to aXb,\, S \to ab,\, aX \to aaXb,\, aX \to ac\}$$

Wir erinnern an die Definition von L_m^n, wobei $m \in \mathbb{N}, n \in \mathbb{N}_{>0}$:

$$\boxed{L_m^n \quad := \{w \in (V \cup \Sigma)^* \mid |w| \leq n \text{ und } S \Rightarrow_G^k w \text{ mit } k \leq m\}}$$

L_m^n ist also die Menge aller Satzformen der Länge höchstens n, die in höchstens m Schritten von S aus ableitbar sind.

a) Berechnen Sie L_i^5 für alle $i \in \mathbb{N}$. Begründen Sie kurz Ihre Berechnung. Die Berechnung von L_0^5 und L_1^5 ist bereits angegeben.

$L_0^5 = \{S\}$

$L_1^5 = \{S\} \cup \{aXb, ab\}$ weil $S \to aXb \in P$ und $S \to ab \in P$

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 3:

b) Geben Sie anhand der obigen Berechnung zwei Wörter an, die in der Sprache $L(G_3)$ liegen.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 3:

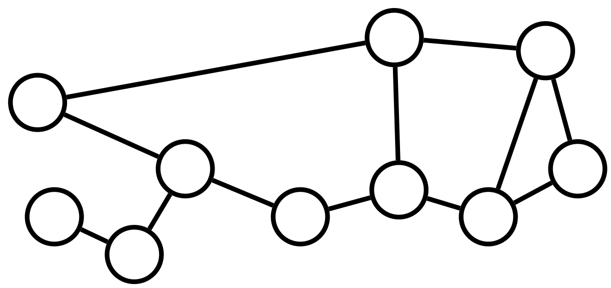
c) Geben Sie anhand der obigen Berechnung zwei Wörter an, die **nicht** in der Sprache $L(G_3)$ liegen.

Aufgabe 4 (Komplexitätstheorie): (16 Punkte)

Wir erinnern zunächst an die Definition des GRAPH-COLORING-Problems:

gegeben:	Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$
gefragt:	Gibt es eine Färbung der Knoten in V mit $\leq k$ Farben, sodass keine zwei benachbarten Knoten in G die gleiche Farbe erhalten?

a) Es sei G_1 der folgende Graph und $k = 3$:



Zeigen Sie, dass $(G_1, k) \in$ GRAPH-COLORING gilt. In Ihrer Antwort können Sie einzelne Buchstaben als Namen für die Farben (z.B. R für Rot) nutzen.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 4:

b) Es sei G_1 derselbe Graph wie in Frage a) und $k = 2$. Zeigen Sie, dass $(G_1, k) \notin$ GRAPH-COLORING gilt.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 4:

c) Der folgende Beweis ist falsch. Lokalisieren Sie den Fehler und begründen Sie, wieso der Beweis falsch ist.

> **Satz:** GRAPH-COLORING ist $\mathcal{NP}$-schwer.
>
> *Beweis:* Wir zeigen GRAPH-COLORING $\leq_{\mathrm{p}}$ SAT. Das heißt, wir werden ein beliebiges GRAPH-COLORING-Problem als SAT-Problem kodieren. Da wir wissen, dass SAT $\mathcal{NP}$-schwer ist, zeigt dies, dass GRAPH-COLORING $\mathcal{NP}$-schwer ist.
>
> Sei $G = (V, E)$ ein Graph. Für jeden Knoten $v \in V$ stellen wir k Variablen auf, eine pro Farbe: $v_1, \ldots, v_k$. Die Variable v_i ist wahr, wenn v die i-te Farbe hat.
>
> Für jeden Knoten v erzeugen wir zuerst die aussagenlogische Formel $v_1 \vee \cdots \vee v_k$, die sicherstellt, dass jeder Knoten mindestens eine Farbe hat.
>
> Zusätzlich erzeugen wir die Formeln $\neg v_i \vee \neg v_j$ für alle $i, j \in \{1, \ldots, k\}$, sodass $i \neq j$. Zusammen stellen diese Formeln sicher, dass jeder Knoten maximal eine Farbe hat.
>
> Ferner erzeugen wir für jedes Paar von benachbarten Knoten v, w und jede Farbe $i \in \{1, \ldots, k\}$ die Formel $\neg v_i \vee \neg w_i$, die sicherstellt, dass benachbarte Knoten nicht die gleiche Farbe haben.
>
> Schlussendlich nehmen wir als Formel F die Konjunktion $F_1 \wedge \cdots \wedge F_n$ von allen oben beschriebenen Formeln.
>
> Damit gilt: Die Formel F ist genau dann erfüllbar, wenn G k-färbbar ist. Da die Übersetzung in Polynomialzeit berechnet werden kann, haben wir GRAPH-COLORING $\leq_{\mathrm{p}}$ SAT gezeigt.

Aufgabe 5 (Gemischte Fragen): (12 Punkte)

Pro Frage gibt es **genau eine richtige** Antwort.

Jede Antwort ist 2 Punkte wert. Eine Antwort, bei der 0 oder mindestens 2 Kästchen angekreuzt sind, zählt als falsch.

WHILE-berechenbare Funktionen sind manchmal

- ☐ nicht μ-rekursiv
- ☐ nicht Turing-berechenbar
- ☐ nicht GOTO-berechenbar
- ☐ nicht LOOP-berechenbar

Das INDEPENDENT-SET-Problem ist $\mathcal{NP}$-vollständig. Zudem ist bekannt, dass

- ☐ INDEPENDENT-SET $\in \mathcal{P}$
- ☐ INDEPENDENT-SET $\notin \mathcal{P}$
- ☐ Wenn INDEPENDENT-SET $\in \mathcal{P}$, dann gilt $\mathcal{P} = \mathcal{NP}$
- ☐ Wenn $\mathcal{P} \neq \mathcal{NP}$, dann INDEPENDENT-SET $\in \mathcal{P}$

Sei $L_4 = \{w \mid \text{Die Turingmaschine } M_w \text{ berechnet für jede Eingabe } x \text{ als Ausgabe die Zahl } x+1 \text{ (binär kodiert)}\}$. Die Sprache L_4 ist unentscheidbar aufgrund des Satzes von Rice, denn $C(S) = L_4$ für

- ☐ $S = \emptyset$
- ☐ $S = \{f\}$, wobei $f(x) = x$ für alle x
- ☐ $S = \{f\}$, wobei $f(x) = 1 + x$ für alle x
- ☐ $S = \{f \mid f \text{ ist eine in polynomieller Zeit berechenbare Funktion}\}$

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 5:

Das LOOP-Programm

$$
\begin{array}{l}
x_0 := x_1 + 2; \\
\textbf{LOOP } x_2 \textbf{ DO} \\
\quad x_0 := x_0 + 3 \\
\textbf{END}
\end{array}
$$

berechnet die Funktion f mit

- ☐ $f(x) = 2 \cdot x$
- ☐ $f(x, y) = 3 \cdot y + 2$
- ☐ $f(x, y) = 3 \cdot y + x + 2$
- ☐ $f(x, y, z) = x \cdot y + 5 \cdot z$

Seien L_5 und L_6 kontextfreie Sprachen.

- ☐ Dann gibt es einen DFA, der $L_5 \cup L_6$ akzeptiert.
- ☐ Dann gibt es einen NFA, der $L_5 \cup L_6$ akzeptiert.
- ☐ Dann gibt es einen Kellerautomaten, der $L_5 \cup L_6$ akzeptiert.
- ☐ Dann gibt es einen regulären Ausdruck, der $L_5 \cup L_6$ erzeugt.

Die Grammatik $G = (\{A, B, C, D\}, \{a, b\}, P, A)$ mit

$$
\begin{aligned}
P = \{ & A \to a, \\
& A \to BC, \\
& B \to b, \\
& B \to CC, \\
& D \to b, \\
& D \to BC\}
\end{aligned}
$$

- ☐ ist vom Typ 0, aber nicht vom Typ 1
- ☐ ist vom Typ 1, aber nicht vom Typ 2
- ☐ ist vom Typ 2, aber nicht vom Typ 3
- ☐ ist vom Typ 3
