Lehrstuhl für Theoretische Informatik
School of Computation, Information and Technology
Technische Universität München

Esolution
Sticker mit SRID hier einkleben

Hinweise zur Personalisierung:

- Ihre Prüfung wird bei der Anwesenheitskontrolle durch Aufkleben eines Codes personalisiert.
- Dieser enthält lediglich eine fortlaufende Nummer, welche auch auf der Anwesenheitsliste neben dem Unterschriftenfeld vermerkt ist.
- Diese wird als Pseudonym verwendet, um eine eindeutige Zuordnung Ihrer Prüfung zu ermöglichen.

Einführung in die Theoretische Informatik

Klausur:	IN0011 / Retake	**Datum:**	Dienstag, 8. Oktober 2024
Prüfer:	Prof. Javier Esparza Philipp Czerner	**Uhrzeit:**	11:30 – 14:30

Bearbeitungshinweise

- Diese Klausur umfasst **16 Seiten** mit insgesamt **8 Aufgaben**.
  Bitte kontrollieren Sie jetzt, dass Sie eine vollständige Angabe erhalten haben.
- Die Gesamtpunktzahl in dieser Klausur beträgt 105 Punkte, von denen 5 Bonuspunkte sind, d.h. sie zählen wie normale Punkte, werden bei der Berechnung des Notenschemas jedoch außer Acht gelassen.
- 45 Punkte sind *hinreichend* zum Bestehen.
- Das Heraustrennen von Seiten aus der Prüfung ist untersagt.
- Als Hilfsmittel sind zugelassen:
  - ein **beidseitig handschriftlich beschriebenes DIN A4 Blatt**
  - ein **analoges Wörterbuch** Deutsch $\leftrightarrow$ Muttersprache **ohne Anmerkungen**
- Mit * gekennzeichnete Teilaufgaben sind ohne Kenntnis der Ergebnisse vorheriger Teilaufgaben lösbar.
- Schreiben Sie weder mit roter / grüner Farbe noch mit Bleistift.
- Schalten Sie alle mitgeführten elektronischen Geräte vollständig aus, verstauen Sie diese in Ihrer Tasche und verschließen Sie diese.
- $\mathbb{N} := \{0, 1, 2, \dots\}$

Hörsaal verlassen von _____ bis _____ / Vorzeitige Abgabe um _____

Aufgabe 1 Quiz: Reguläre und kontextfreie Sprachen (16 Punkte)

Teilaufgaben (a-e): Für diese Fragen ist eine Begründung nicht gefordert. Sie erhalten die Punkte auf eine Teilaufgabe genau dann, wenn Sie alle Antwortmöglichkeiten korrekt angekreuzt haben. **Es ist immer mindestens eine Antwortmöglichkeit richtig.** Jede Frage bringt 2P.

Kreuzen Sie richtige Antworten an ☒
Kreuze können durch vollständiges Ausfüllen gestrichen werden ■
Gestrichene Antworten können durch nebenstehende Markierung erneut angekreuzt werden ×■

In dieser Aufgabe verwenden wir durchgehend das Alphabet $\Sigma := \{a, b\}$.

a)* Welche der folgenden Sprachen sind Residualsprachen von $L((bba)^*b)$?

☐ $\{b\}$ ☒ $L(ba(bba)^*b \mid \epsilon)$ ☒ $\emptyset$

Die Residualsprachen sind $\emptyset$, $L(ba(bba)^*b \mid \epsilon)$, $L(a(bba)^*b)$, und die Sprache selbst.

b)* Sei $L \subseteq \Sigma^*$ eine Sprache mit $LL = \Sigma^*$. Welche Aussagen sind wahr?

☒ L ist unendlich ☒ $\varepsilon \in L$ ☒ $LLL = \Sigma^*$

(1) Wenn L endlich wäre, wäre LL auch endlich. (2) Da $\varepsilon \in \Sigma^*$, muss auch $\varepsilon \in LL$ gelten, also $\varepsilon \in L$. (3) Es gilt $LLL = \Sigma^* L \supseteq \Sigma^*$ wegen $\varepsilon \in L$.

c)* Sei $L \subseteq \Sigma^*$ regulär. Welche der folgenden Sprachen sind regulär?

☒ $\{u^R v \mid u, v \in L\}$ ☐ $\{u^R u \mid u \in L\}$ ☐ $\{uv \mid u, v \in L \wedge |u| = |v|\}$

(1) Das ist $L^R L$, Abgeschlossenheit unter Konkatenation. (2) Gegenbeispiel: $L = \Sigma^*$. (3) Gegenbeispiel: $L = L(ba^*)$. Dann ist diese Sprache $\{ba^n ba^n \mid n \in \mathbb{N}\}$, was nicht regulär ist.

d)* Welche der folgenden kontextfreien Grammatiken sind mehrdeutig?

☒ $S \to SS \mid a$ ☐ $S \to Sa \mid Sb \mid \varepsilon$ ☒ $S \to TT \quad T \to aT \mid bT \mid a \mid b$

(1), (3) aaa hat zwei Syntaxbäume. (2) Die Terminale werden von rechts nach links erzeugt.

e)* Sei $A \subseteq \Sigma^*$ regulär und $B \subseteq \Sigma^*$ kontextfrei. Welche der folgenden Sprachen sind kontextfrei?

☐ $A \cup \overline{B}$ ☒ $\overline{A} \cup B$ ☒ AB

(1) Gegenbeispiel: $A = \emptyset$, $B = \overline{\{a^n b^n a^n \mid n \geq 0\}}$. (2) $\overline{A}$ ist nach Abschlusseigenschaften regulär, also kontextfrei, Abgeschlossenheit unter Vereinigung. (3) A ist kontextfrei, Abgeschlossenheit unter Konkatenation.

0 1 2 3

f)* Geben Sie reguläre Ausdrücke X, Y, Z über Σ an, sodass die folgenden Gleichungen erfüllt sind.

$$X \equiv a^*b^*aX \mid ab \qquad Y \equiv (b \mid a)a^*Y \mid bbY \mid \epsilon \qquad Z \equiv a^*(a \mid b)Z$$

$X = (a^*b^*a)^*ab$ $Y = ((b \mid a)a^* \mid bb)^*\epsilon$ $Z = \emptyset$

0 1 2 3

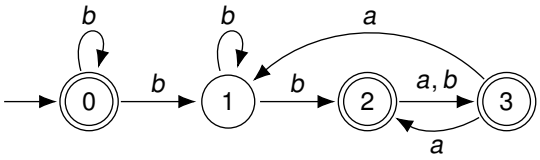
g)* Ein NFA $(Q, \Sigma, \delta, q_0, F)$ heißt *reversibel*, falls für alle Zustände $q_1, q_2 \in Q$ und für alle Zeichen $x \in \Sigma$ gilt: Falls $q_2 \in \delta(q_1, x)$, so $q_1 \in \delta(q_2, x)$. Mit anderen Worten: Falls man von q_1 mit x in q_2 übergehen kann, so kann man auch von q_2 mit x in q_1 übergehen.
Gibt es einen reversiblen NFA N über Σ mit $L(N) = L((aa)^* \mid (bb)^*)$? Falls ja, geben Sie einen solchen NFA an; falls nein, beweisen Sie es.

Ihre Wahl: ☐ Wahr, reversibler NFA ☒ Falsch, Beweis

Eine mögliche Lösung: Sei N ein solcher NFA mit Startzustand q_0. Da ε akzeptiert wird, muss q_0 ein Endzustand sein. Da aa und bb akzeptiert werden, muss q_0 eine ausgehende Kante mit a und b besitzen. Aufgrund der Eigenschaft des reversiblen NFAs kommt man also von q_0 nach q_0 sowohl mit aa als auch mit bb. Dann würde aber auch $aabb$ akzeptiert werden.

Aufgabe 2 DFA-Algorithmen (18 Punkte)

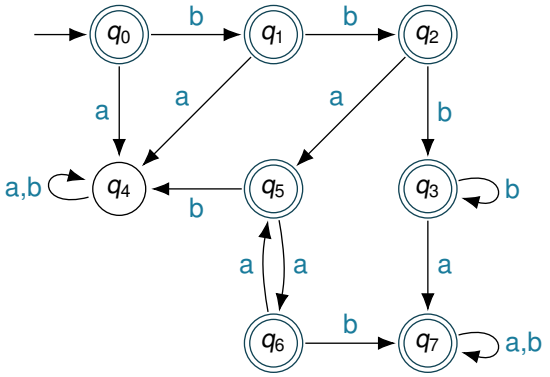
a) Wir betrachten den folgenden NFA N über dem Alphabet $\Sigma = \{a, b\}$:



0 1 2 3 4 5 6

Sei M der DFA, den man erhält, wenn man N mithilfe des Verfahrens aus der Vorlesung (Potenzmengenkonstruktion) zu einem DFA konvertiert. Tragen Sie M in untere Schablone ein, indem Sie (1) die Transitionen beschriften, (2) die Finalzustände markieren, und (3) in der Tabelle rechts für jeden Zustand von M die korrespondierende Menge an Zuständen aus N eintragen.

Hinweis: Sie können den Platz am Ende der Seite verwenden, um den Algorithmus auszuführen.



q_0:	$\{0\}$	q_4:	$\emptyset$
q_1:	$\{0, 1\}$	q_5:	$\{3\}$
q_2:	$\{0, 1, 2\}$	q_6:	$\{1, 2\}$
q_3:	$\{0, 1, 2, 3\}$	q_7:	$\{1, 2, 3\}$

b) Ist M minimal? Begründen Sie Ihre Antwort kurz.

0 1

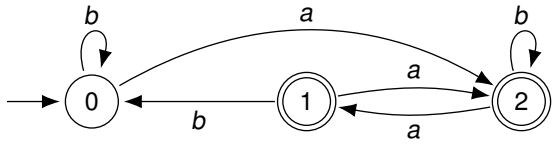
☐ Ja ☒ Nein

Begründung: Die Zustände q_3 und q_7 sind äquivalent, da $L(q_3) = L(q_7) = \Sigma^*$.

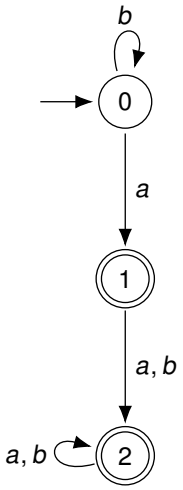
0 1 2 3 4

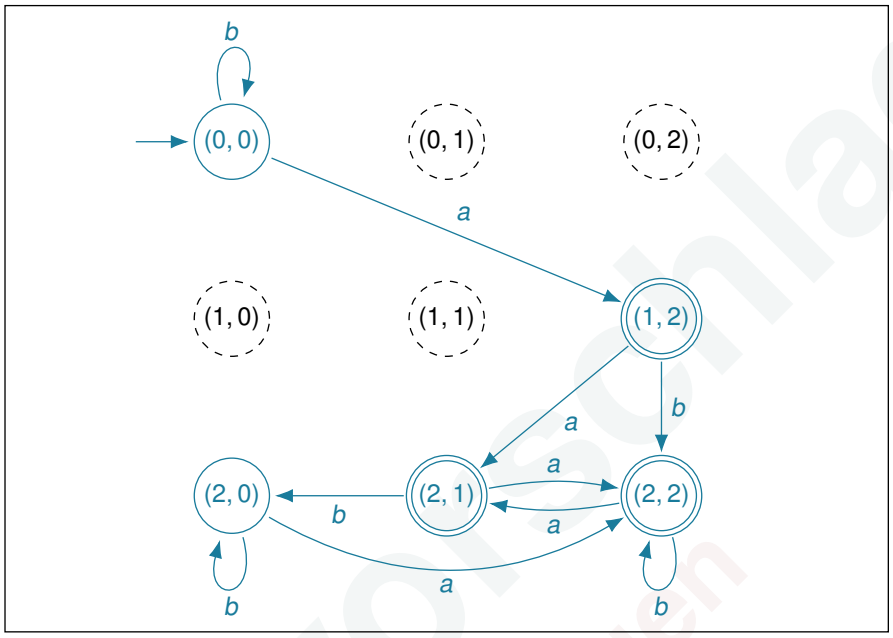
c)* Im Folgenden sind zwei DFAs M_1 und M_2 angegeben. Berechnen Sie nach dem Algorithmus aus der Vorlesung (Produktkonstruktion) einen DFA M mit $L(M) = L(M_1) \cap L(M_2)$ und tragen Sie das Ergebnis unten ein. Platzieren Sie die Zustände an den vorgesehenen Positionen. (Es genügt, die erreichbaren Zustände zu konstruieren.)

M_2:



M_1:





0 1 2

d)* Geben Sie jeweils ein kürzestes Wort in $L(M_1) \cap L(M_2)$, $L(M_1) \setminus L(M_2)$ und $L(M_2) \setminus L(M_1)$ an, oder $\emptyset$, falls kein solches Wort existiert.

Hinweis: Verwenden Sie Ihr Ergebnis aus c).

$L(M_1) \cap L(M_2)$: a $L(M_1) \setminus L(M_2)$: aab $L(M_2) \setminus L(M_1)$: $\emptyset$

0 1 2 3 4 5

e)* Sei $L \subseteq \{a, b, c\}^*$ eine Sprache. Wir definieren die Sprache $R_c(L) \subseteq \{a, b\}^*$ als die Menge der Wörter, die man erhält, wenn man aus jedem Wort von L jedes Zeichen c löscht. Z.B. gilt $R_c(\{bca, cc, ab\}) = \{ba, \varepsilon, ab\}$. Sei M ein DFA mit n Zuständen. Wählen Sie eine Funktion $f : \mathbb{N} \to \mathbb{N}$, sodass es einen DFA M' für $R_c(L(M))$ mit höchstens $f(n)$ Zuständen gibt **und** beschreiben Sie einen Algorithmus, um M' zu berechnen und argumentieren Sie, wieso M' höchstens $f(n)$ Zustände hat.

Ihre Wahl: $f(n) := 2^n$

Wir ersetzen jede c-Transition in M durch eine ϵ-Transition. Somit erhalten wir einen ϵ-NFA für $R_c(L)$ mit n Zuständen. Diesen konvertieren wir zu einem NFA N mit dem Algorithmus aus der Vorlesung, N hat also auch n Zustände. Schließlich konvertieren wir N zu einem DFA M' mit der Potenzmengenkonstruktion aus der Vorlesung, und somit hat M' höchstens 2^n Zustände.

Aufgabe 3 Rekursive Funktionen (13 Punkte)

Sei $\Sigma := \{0,1\}$. Für eine Sprache $L \subseteq \Sigma^*$ bezeichnet $\mathrm{N}(L)$ die Sprache, die man erhält, wenn man den Wörtern aus L **höchstens eine** Null an einer beliebigen Stelle hinzufügt. Formal definieren wir

$$\mathrm{N}(L) := \{vxw \mid v, w \in \Sigma^* \wedge x \in \{0, \varepsilon\} \wedge vw \in L\}$$

a)* Listen Sie alle Wörter der folgenden Sprache explizit auf:

0 1

$\mathrm{N}(\{01, 0\}) =$ $\{01, 001, 010, 0, 00\}$

b)* Geben Sie analog zu den rekursiven Funktionen aus den Übungs- und Hausaufgaben eine rekursive Funktion null an, sodass $\mathrm{null}(r)$ ein regulärer Ausdruck mit $L(\mathrm{null}(r)) = \mathrm{N}(L(r))$ ist, für jeden regulären Ausdruck r. Im Folgenden sind r, r_1, r_2 reguläre Ausdrücke, und $a \in \Sigma$.

0 1 2 3 4 5 6

$\mathrm{null}(\emptyset) :=$ $\emptyset$	$\mathrm{null}(r_1 \mid r_2) :=$ $\mathrm{null}(r_1) \mid \mathrm{null}(r_2)$
$\mathrm{null}(\epsilon) :=$ $\epsilon \mid 0$	$\mathrm{null}(r_1 r_2) :=$ $\mathrm{null}(r_1) r_2 \mid r_1 \mathrm{null}(r_2)$
$\mathrm{null}(a) :=$ $a \mid 0a \mid a0$	$\mathrm{null}(r^*) :=$ $\epsilon \mid 0 \mid r^* \mathrm{null}(r) r^*$, oder $r^*(\epsilon \mid 0 \mid \mathrm{null}(r)) r^*$

c)* Gegeben sei die folgende rekursive Funktion über dem Alphabet $\Sigma := \{0,1\}$:

0 1 2 3 4 5 6

$$\mathrm{cut}(\emptyset) := \emptyset \qquad \mathrm{cut}(0) := \emptyset \qquad \mathrm{cut}(r_1 \mid r_2) := \mathrm{cut}(r_1) \mid \mathrm{cut}(r_2) \qquad \mathrm{cut}(r^*) := r^* \mathrm{cut}(r)$$
$$\mathrm{cut}(\epsilon) := \emptyset \qquad \mathrm{cut}(1) := \epsilon \qquad \mathrm{cut}(r_1 r_2) := \mathrm{cut}(r_1) \mid r_1 \mathrm{cut}(r_2)$$

Per struktureller Induktion über r lässt sich zeigen, dass für alle $u, v \in \Sigma^*$ mit $u1v \in L(r)$ gilt, dass $u \in L(\mathrm{cut}(r))$. Vervollständigen Sie den Beweis für den Fall $r = r_1 r_2$. Ergänzen Sie die Induktionshypothese(n) und führen Sie dann den Beweis. Kennzeichnen Sie Verwendungen der Induktionshypothesen im Beweis deutlich.

Zu zeigen: Für alle $u1v \in L(r_1 r_2)$ mit $u, v \in \Sigma^*$ gilt $u \in L(\mathrm{cut}(r_1 r_2))$.

Induktionshypothese(n) für den Fall $r = r_1 r_2$:

Für $i \in \{1,2\}$ und alle $u1v \in L(r_i)$ mit $u, v \in \Sigma^*$ gilt $u \in L(\mathrm{cut}(r_i))$.

Beweis des Falls $r = r_1 r_2$:

Sei $u1v \in L(r_1 r_2)$ beliebig, mit $u, v \in \Sigma^*$. Da $u1v \in L(r_1 r_2)$, existieren $y \in L(r_1)$ und $z \in L(r_2)$ mit $yz = u1v$.

- Fall 1: $|u1| \leq |y|$. Dann gibt es ein $v' \in \Sigma^*$ mit $u1v' = y$. Nach IH folgt $u \in L(\mathrm{cut}(r_1))$. Somit $u \in L(\mathrm{cut}(r_1 r_2))$ (nach Definition von $\mathrm{cut}(r_1 r_2)$).
- Fall 2: $|u1| > |y|$. Dann gibt es ein $u' \in \Sigma^*$ mit $u'1v = z$ und $yu' = u$. Nach IH folgt $u' \in L(\mathrm{cut}(r_2))$, und mit $y \in L(r_1)$ gilt $u = yu' \in L(r_1 \mathrm{cut}(r_2))$. Wieder $u \in \mathrm{cut}(r_1 r_2)$ nach Definition von $\mathrm{cut}(r_1 r_2)$.

Aufgabe 4 CFG-Algorithmen (13 Punkte)

In den Teilaufgaben (a)-(d) geht es darum, eine kontextfreie Grammatik (CFG) in Chomsky-Normalform (CNF) zu konvertieren. Wir führen jeden Schritt einzeln aus, und jeweils auf einer anderen Grammatik – Sie können also die Aufgabenteile unabhängig voneinander bearbeiten.
Eine CFG $G = (V, \Sigma, P, S)$ ist in CNF, wenn jede Produktion $(X \to \alpha) \in P$, mit $X \in V$ und $\alpha \in (\Sigma \cup V)^*$, folgende Bedingungen erfüllt:

(1) $\alpha \in \Sigma \cup V^*$; Terminale dürfen nur in Produktionen der Länge 1 erzeugt werden.

(2) $|\alpha| \leq 2$; jede Produktion hat höchstens Länge 2.

(3) $\alpha \neq \varepsilon$; es gibt keine ε-Produktionen.

(4) $\alpha \notin V$; es gibt keine Kettenproduktionen.

Achtung: Die Teilaufgaben fragen jeweils spezifisch nach dem Ergebnis, das sich durch die Ausführung des Algorithmus aus der Vorlesung ergibt, nicht nach einer beliebigen äquivalenten CFG, die den Bedingungen genügt. Details, wie etwa die Namen der Variablen oder die Reihenfolge, in der Produktionen betrachtet werden, können Sie frei wählen.

0 1 2

a)* *Entfernen von Terminalen in langen Produktionen.* Die CFG G_a ist gegeben durch folgende Produktionen:

$$S \to aX \mid Xa \mid ab \qquad X \to SS \mid S \mid \varepsilon \mid bXX$$

Führen Sie den ersten Schritt des Algorithmus zur Überführung in CNF aus und geben Sie die Produktionen einer CFG G'_a an, so dass $L(G_a) = L(G'_a)$ gilt und G'_a Bedingung (1) erfüllt.

(Platz für Zwischenschritte / Erklärungen)

Fertige Grammatik G'_a:

$S \to AX \mid XA \mid AB$
$X \to SS \mid S \mid \varepsilon \mid BXX$
$A \to a$
$B \to b$

0 1 2

b)* *Entfernen langer Produktionen.* Die CFG G_b ist gegeben durch die folgenden Produktionen:

$$S \to A \mid BAA \mid \varepsilon \qquad A \to SABB \mid a \qquad B \to AS \mid b$$

Führen Sie den zweiten Schritt des Algorithmus zur Überführung in CNF aus und geben Sie die Produktionen einer CFG G'_b an, so dass $L(G_b) = L(G'_b)$ und G'_b Bedingungen (1) und (2) erfüllt.

(Platz für Zwischenschritte / Erklärungen)

Fertige Grammatik G'_b:

$S \to A \mid BX_{AA} \mid \varepsilon$ $\quad X_{AA} \to AA$
$A \to SX_{ABB} \mid a$ $\quad X_{ABB} \to AX_{BB}$
$B \to AS \mid b$ $\quad X_{BB} \to BB$

c)* *Entfernen von ε-Produktionen.* Die CFG G_c ist gegeben durch folgende Produktionen:

0 1 2 3

$$S \to AB \mid DE \qquad C \to A \mid c$$
$$A \to AB \mid a \mid \varepsilon \qquad D \to BC \mid BS$$
$$B \to b \mid BB \mid D \qquad E \to SS \mid CA$$

Führen Sie den dritten Schritt des Algorithmus zur Überführung in CNF aus. Geben Sie die Produktionen einer CFG G_c' an, so dass $L(G_c) = L(G_c')$ und G_c' Bedingungen (1), (2) und (3) erfüllt.

(Platz für Zwischenschritte / Erklärungen)

Die Nichtterminale, die ε erzeugen, berechnen wir iterativ:
1. $\{A\}$, 2. $\{A, C\}$, 3. $\{A, C, E\}$.
Dann fügen wir die entsprechenden neuen Produktionen ein und entfernen die ε-Produktionen:

Fertige Grammatik G_c':

$S \to AB \mid DE \mid B \mid D$
$A \to AB \mid a \mid B$
$B \to b \mid BB \mid D$
$C \to A \mid c$
$D \to BC \mid BS \mid B$
$E \to SS \mid CA \mid C \mid A$

d)* *Entfernen von Kettenproduktionen.* Die CFG G_d ist gegeben durch die Produktionen:

0 1 2 3

$$S \to a \mid BA \mid B \qquad B \to b \mid SC \mid A$$
$$A \to a \mid AA \qquad C \to B \mid SS$$

Führen Sie den vierten Schritt des Algorithmus zur Überführung in CNF aus und geben Sie die Produktionen einer CFG G_d' in CNF an, so dass $L(G_d) = L(G_d')$ gilt.

(Platz für Zwischenschritte / Erklärungen)

Die Variablen, die wir mit Kettenproduktionen ineinander umwandeln können, berechnen wir iterativ:
1. $S \to B, B \to A, C \to B$
2. zusätzlich $S \to A, C \to A$
Dann entfernen wir die Kettenproduktionen und fügen stattdessen entsprechende neue Produktionen ein:

Fertige Grammatik G_d':

$S \to a \mid BA \mid b \mid SC \mid AA$
$A \to a \mid AA$
$B \to b \mid SC \mid a \mid AA$
$C \to b \mid SC \mid a \mid AA \mid SS$

e)* Die CFG G ist gegeben durch die Produktionen:

0 1 2 3

$$S \to AB \mid C \qquad A \to aA \mid AS \qquad B \to bS \qquad C \to aCb \mid \varepsilon$$

Geben Sie die erzeugenden, erreichbaren und nützlichen Nichtterminale von G an.

Erzeugend: S, B, C Erreichbar: S, A, B, C Nützlich: S, C

Aufgabe 5 Verlorene Buchsten (12 Punkte)

Sei $\Sigma := \{a, b\}$. Für eine Sprache $L \subseteq \Sigma^*$ bezeichnet $\text{del}(L)$ die Sprache, die man erhält, wenn man den Wörtern aus L **genau einmal** das Wort ab an einer beliebigen Stelle entfernt. Formal definieren wir

$$\text{del}(L) := \{vw \mid v, w \in \Sigma^* \wedge vabw \in L\}$$

0 1

a)* Listen Sie alle Wörter der folgenden Sprache explizit auf:

$\text{del}(\{abbaba, baa, ab, a, b\}) =$ $\{baba, abba, \varepsilon\}$

0 1 2

b)* Wir definieren $\text{del2}(L) := \{uvw \mid u, v, w \in \Sigma^* \wedge uabvabw \in L\}$. Zeigen Sie oder widerlegen Sie mit einem Gegenbeispiel: Für jede **endliche** Sprache $L \subseteq \Sigma^*$ gilt $\text{del}(\text{del}(L)) = \text{del2}(L)$.

Ihre Wahl: ☐ Wahr ☒ Falsch

Beweis: Betrachte $L := \{aabb\}$. Dann gilt $\text{del}(\text{del}(L)) = \{\varepsilon\} \neq \emptyset = \text{del2}(L)$.

0 1 2 3 4 5 6 7 8 9

c)* Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik in Chomsky-Normalform. Wir möchten eine kontextfreie Grammatik $G' = (V', \Sigma, P', S')$ konstruieren, sodass $L(G') = \text{del}(L(G))$ gilt. Hierzu setzen wir

$$V' := V \cup \{X_{ab} \mid X \in V\} \cup \{X_a \mid X \in V\} \cup \{X_b \mid X \in V\},$$

wobei $X_{ab}, X_a, X_b \notin V$ für alle $X \in V$. Definieren Sie nun P' und S', sodass

1. $L_{G'}(X) = L_G(X)$
2. $L_{G'}(X_{ab}) = \text{del}(L_G(X))$
3. $L_{G'}(X_a) = \{w \in \Sigma^* \mid wa \in L_G(X)\}$
4. $L_{G'}(X_b) = \{w \in \Sigma^* \mid bw \in L_G(X)\}$

für alle $X \in V$ gilt. Geben Sie die Produktionen P' präzise an; vermeiden Sie unstrukturierten Fließtext.
Hinweis: G ist in CNF, hat also keine ε-Produktionen. G' muss nicht in CNF sein.

Startsymbol $S' := S_{ab}$

Produktionen P':

$X \to \alpha,$	für $X \to \alpha \in P$
$X_{ab} \to Y_{ab}Z \mid YZ_{ab} \mid Y_aZ_b,$	für $X \to YZ \in P$
$X_a \to YZ_a,$	für $X \to YZ \in P$
$X_a \to \varepsilon,$	für $X \to a \in P$
$X_b \to Y_bZ,$	für $X \to YZ \in P$
$X_b \to \varepsilon,$	für $X \to b \in P$

Wenden Sie Ihre Konstruktion auf das Beispiel $G := (\{S, A\}, \{a, b\}, P, S)$ an, wobei P gegeben ist durch:

$$S \to SA \mid a \qquad A \to a \mid b$$

Hinweis: Falls ein Nichtterminal nicht auf der linken Seite einer Produktion vorkommt, lassen Sie das entsprechende Feld leer.

Startsymbol $S' := S_{ab}$

Produktionen P':

$S \to SA \mid a$	$A \to a \mid b$
$S_{ab} \to S_{ab}A \mid SA_{ab} \mid S_aA_b$	$A_{ab} \to$
$S_a \to SA_a \mid \varepsilon$	$A_a \to \varepsilon$
$S_b \to S_bA$	$A_b \to \varepsilon$

Aufgabe 6 Quiz: Berechenbarkeit und Komplexität (14 Punkte)

Teilaufgaben (a-e): Für diese Fragen ist eine Begründung nicht gefordert. Sie erhalten die Punkte auf eine Teilaufgabe genau dann, wenn Sie alle Antwortmöglichkeiten korrekt angekreuzt haben. **Es ist immer mindestens eine Antwortmöglichkeit richtig.** Jede Frage bringt 2P.

In dieser Aufgabe verwenden wir durchgehend das Alphabet $\Sigma := \{0, 1\}$.

a)* Sei L eine kontextfreie Sprache. Welche Aussagen sind wahr?

☒ L ist entscheidbar ☒ $\overline{L}$ ist entscheidbar ☒ L ist semi-entscheidbar

(1) folgt z.B. aus Existenz von CYK, (2) und (3) dann direkt aus (1).

b)* Seien $L_1, L_2 \subseteq \Sigma^*$ entscheidbare Sprachen. Welche der folgenden Sprachen sind entscheidbar?

☒ $L_1 \cap L_2$ ☒ $L_1 \setminus L_2$ ☒ $L_1 L_2$

(1,2) Abschlusseigenschaften. (3) Auf Eingabe w gehen wir alle Zerlegungen in zwei Teilwörter durch.

c)* Sei $L \subseteq \{0\}^*$ unentscheidbar. Welche der folgenden Funktionen $f : \mathbb{N} \to \mathbb{N}$ sind berechenbar?

☐ $f(n) := |L \cap \Sigma^n|$ ☒ $f(n) := \min\{k \in \mathbb{N} : 0^k \in L\}$ ☐ $f(n) := 1$, falls $0^n \in L$, sonst $f(n) := 0$

(1,3) sonst ließe sich L entscheiden. (2) f ist konstant, also berechenbar.

d)* Seien $A, B \subseteq \Sigma^*$ Sprachen mit $A \leq_p \text{SAT} \leq_p B$. Welche Aussagen sind wahr?

☒ $A \in$ NP ☐ $B \in$ NP ☒ B ist NP-hart ☒ $A \leq_p B$

(1,3) da SAT NP-vollständig ist. (2) $\mathcal{H}_0$ ist ein Gegenbeispiel. (4) Reduktionsfunktionen verketten, s. H12.3c.

e)* Sei $L \subseteq \Sigma^*\{\#\}\Sigma^*$ eine Sprache, sodass $L \in$ P; L enthält also Wörter der Form $u\#v$, mit $u, v \in \Sigma^*$. Welche der folgenden Sprachen sind in NP?

☒ L ☐ $\{u \mid \exists v \in \Sigma^* : u\#v \in L\}$ ☒ $\{u \mid \exists v \in \Sigma^* : u\#v \in L \wedge |u| = |v|\}$

(1) P $\subseteq$ NP, (2) Gegenbeispiel $L := \{G\#w \mid G \text{ CFG} \wedge w \notin L(G)\}$, (3) die NTM kann v raten.

0 1 2 3 4

f)* Sei $L \in$ NP. Zeigen oder widerlegen Sie: $L^* \in$ NP.

☒ Ja, Beweis: ☐ Nein, Gegenbeispiel:

Sei N eine NTM, die L in polynomieller Zeit entscheidet. Auf Eingabe $w \in \Sigma^*$ raten wir eine Zerlegung $w = w_1 \cdots w_n$, wobei $w_1, ..., w_n \in \Sigma^*$ Wörter sind. Nun akzeptieren wir, falls $w_i \in L$ für alle $i \in \{1, ..., n\}$ gilt, was wir überprüfen, indem wir N auf diesen Eingaben ausführen.

Aufgabe 7 **Reduktion (14 Punkte)**

Nehmen Sie P $\neq$ NP an. Entscheiden Sie unter dieser Annahme, welche der folgenden Aussagen gelten. Die Aussagen sind von der Gestalt „$A \leq_p B$“ für Entscheidungsprobleme A, B. Wenn die Aussage gilt, dann müssen Sie (1) eine präzise definierte Reduktionsfunktion angeben **und** (2) die Reduktionsfunktion auf die gegebene Instanz von Problem A anwenden. Die Korrektheit der Reduktion müssen Sie nur skizzieren. Wenn die Aussage nicht gilt, dann müssen Sie dies beweisen.

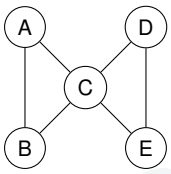
a)* Sei $G = (V, E)$ ein ungerichteter Graph. Ein Hamilton-Pfad von G ist eine Permutation $v_1, \dots, v_n$ der Knoten von V mit $\{v_i, v_{i+1}\} \in E$ für jedes $i \in \{1, \dots, n-1\}$. Ein Hamilton-Kreis von G ist ein Hamilton-Pfad von G, der zusätzlich $\{v_n, v_1\} \in E$ erfüllt.
Sei HAMILTON-PFAD das Entscheidungsproblem, ob ein gegebener ungerichteter Graph einen Hamilton-Pfad hat. Sei HAMILTON-KREIS das Entscheidungsproblem, ob ein gegebener ungerichteter Graph einen Hamilton-Kreis hat.
Aussage: HAMILTON-PFAD $\leq_p$ HAMILTON-KREIS

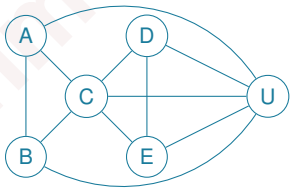
0 1 2 3 4

Ihre Wahl: ☒ Die Aussage gilt ☐ Die Aussage gilt nicht

Reduktionsfunktion: Sei $G = (V, E)$. Wir definieren $f(G) = (V', E')$ mit $V' := V \cup \{u\}$, wobei $u \notin V$, und $E' := E \cup \{\{u, v\} \mid v \in V\}$.
Korrektheitsbeweis: Wenn $v_1, \dots, v_n$ ein Hamilton-Pfad von G ist, dann ist $v_1, \dots, v_n, v$ ein Hamilton-Kreis von G'. Wenn o.B.d.A. $v_1, \dots, v_n, v$ ein Hamilton-Kreis von G' ist, dann ist $v_1, \dots, v_n$ ein Hamilton-Pfad von G.

Falls die Aussage gilt: Wenden Sie Ihre Reduktion auf folgende Instanz von HAMILTON-PFAD an.





b)* Sei CFG-LEERHEIT das Entscheidungsproblem, ob eine gegebene kontextfreie Grammatik G kein Wort erzeugt, also ob $L(G) = \emptyset$ gilt.
Aussage: 3KNF-SAT $\leq_p$ CFG-LEERHEIT

0 1 2 3

Ihre Wahl: ☐ Die Aussage gilt ☒ Die Aussage gilt nicht

Nach Vorlesung ist CFG-LEERHEIT in P und 3KNF-SAT ist NP-vollständig. Wenn 3KNF-SAT $\leq_p$ CFG-LEERHEIT, dann gilt 3KNF-SAT $\in$ P und damit P = NP, Widerspruch.

Falls die Aussage gilt: Wenden Sie Ihre Reduktion auf folgende Instanz von 3KNF-SAT an.

$$(x_1 \vee \neg x_2 \vee x_3) \\ \wedge (\neg x_1 \vee x_2 \vee x_3)$$

0 1 2 3 4 5 6 7

c)* EXAKTE-ÜBERDECKUNG ist das folgende Entscheidungsproblem:
Eingabe: Eine endliche Menge $M = \{m_1, \dots, m_k\}$ und eine Menge $T = \{T_1, \dots, T_n\}$ von Teilmengen von M.
Ausgabe: Enthält T eine *exakte Überdeckung* von M, d.h. gibt es eine Menge $S \subseteq T$, sodass jedes Element von M in **genau** einer der Teilmengen in S vorkommt?
Aussage: EXAKTE-ÜBERDECKUNG $\leq_p$ SAT

Ihre Wahl: ☒ Die Aussage gilt ☐ Die Aussage gilt nicht

Für jedes $m \in M$, definiere $\mathit{Enthält}_m := \{T_i \mid m \in T_i\}$.
Reduktionsfunktion f: Seien $\{t_1, \dots, t_n\}$ boolesche Variablen. (Intuition: t_i ist wahr bedeutet, dass $T_i \in S$.) Wir setzen

$$f(M, T) := \bigwedge_{m \in M} (\mathit{Mindestens\_eine}_m \wedge \mathit{Höchstens\_eine}_m)$$

mit

$$\mathit{Mindestens\_eine}_m := \bigvee_{T_i \in \mathit{Enthält}_m} t_i$$

(mindestens eine der Teilmengen, die m enthalten, gehört zu S)

$$\mathit{Höchstens\_eine}_m := \bigwedge_{i<j, T_i, T_j \in \mathit{Enthält}_m} (\neg t_i \vee \neg t_j)$$

(höchstens eine der Teilmengen, die m enthalten, gehört zu S).
Eine Belegung σ von $f(M, T)$ ist erfüllend gdw. die Menge $S := \{T_i \mid \sigma(t_i) = 1\}$ eine exakte Überdeckung von M ist.

Falls die Aussage gilt: Wenden Sie Ihre Reduktion auf folgende Instanz von EXAKTE-ÜBERDECKUNG an.

$M := \{1, 2, 3, 4\}$ und
$T := \{T_1, T_2, T_3, T_4, T_5\}$,
wobei
$T_1 := \{1\}$, $T_2 := \{1, 3\}$,
$T_3 := \{1, 2, 3\}$,
$T_4 := \{2, 4\}$, $T_5 := \emptyset$.

$$\Big((t_1 \vee t_2 \vee t_3) \wedge ((\neg t_1 \vee \neg t_2) \wedge (\neg t_1 \vee \neg t_3) \wedge (\neg t_2 \vee \neg t_3))\Big) \\ \wedge \Big((t_3 \vee t_4) \wedge ((\neg t_3 \vee \neg t_4))\Big) \\ \wedge \Big((t_2 \vee t_3) \wedge ((\neg t_2 \vee \neg t_3))\Big) \\ \wedge \Big((t_4) \wedge \mathrm{True}\Big)$$

Aufgabe 8 **Bonus (5 Punkte)**

Die Punkte dieser Aufgabe sind **Bonuspunkte**, d.h. sie zählen wie normale Punkte, werden bei der Berechnung des Notenschemas jedoch außer Acht gelassen.
Mindestens eine Antwortmöglichkeit ist immer richtig.

a)* Sei $\Sigma := \{a, b, c\}$. Welche der folgenden Sprachen sind **NICHT** regulär? Kreuzen Sie **alle** zutreffenden an. Wähle Sie anschließend **eine** solche Sprache und beweisen Sie, dass diese nicht regulär ist.

0 1 2

☐ $\{w \in \Sigma^* : |w|_{aba} \neq |w|_a\}$ ☐ $\left(\{w \in \Sigma^* : w = w^R\}\right)^*$ ☒ $\{a^i b^j c^k : (i \neq 1 \vee j = k) \wedge i, j, k \geq 0\}$

Beweis für Sprache 3 (1, 2 oder 3):
Sei L_3 die Sprache. Falls L_3 regulär wäre, dann nach Abschlusseigenschaften auch $L' := L_3 \cap L(ab^*c^*) = \{ab^n c^n : n \in \mathbb{N}\}$, und somit auch $(L')^a = \{b^n c^n : n \in \mathbb{N}\}$, aber diese Sprache entspricht einer bekannten nicht-regulären Sprache, ein Widerspruch.

Anmerkung: L_1 ist einfach die Sprache der Wörter, die ein a enthalten. L_2 ist einfach Σ^*, da $a, b, c \in L_2$.

b)* Sei $\Sigma := \{a, b\}$. Für welche der folgenden Sprachen L gilt $L \neq L^R$? Geben Sie außerdem für jede Sprache L, die Sie gewählt haben, ein Wort w an, das in genau einer der Sprachen L, L^R enthalten ist.

0 1

☒ $L(b^*a^*b)$
Wort, falls $L \neq L^R$:
ab

☐ $\{w \in \Sigma^* : w \neq w^R\}$
Wort, falls $L \neq L^R$:

☒ $\{uavv : u, v \in \Sigma^*\}$
Wort, falls $L \neq L^R$:
babb

c)* Welche der folgenden Sprachen über $\Sigma := \{0, 1\}$ sind unentscheidbar? Kreuzen Sie **alle** zutreffenden an. Wähle Sie anschließend **eine** solche Sprache und beweisen Sie, dass sie unentscheidbar ist.

0 1 2

☒ $\{w \in \Sigma^* : |L(M_w)|^2 = 25\}$ ☐ $\{w \in \Sigma^* : |L(M_w)|^2 = 50\}$ ☒ $\{w \in \Sigma^* : |L(M_w)|^2 = 100\}$

Beweis für Sprache 1 (1, 2 oder 3):
Wir verwenden den Satz von Rice. Sei F die Menge aller berechenbaren Funktionen über Σ, die genau auf 5 Eingaben definiert sind. F ist offensichtlich nicht trivial (es gibt Funktionen, die auf mehr bzw. weniger als 5 Eingaben definiert sind). Somit ist $\{w \in \Sigma^* : |L(M_w)|^2 = 25\} = \{w \in \Sigma^* : \varphi_w \in F\}$ unentscheidbar.

Anmerkung: Die zweite Sprache ist leer, da 50 keine Quadratzahl ist.

Zusätzlicher Platz für Lösungen. Markieren Sie deutlich die Zuordnung zur jeweiligen Teilaufgabe. Vergessen Sie nicht, ungültige Lösungen zu streichen.