1. Klausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2021/2022

Hier Aufkleber mit Name und Matrikelnummer anbringen

Vorname: _____

Nachname: _____

Matrikelnummer: _____

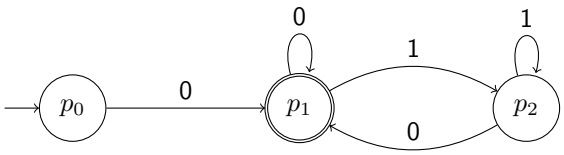
- Bringen Sie den Aufkleber mit Ihrem Namen und Ihrer Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Am Ende der Klausur sind zusätzliche Leerseiten. Fordern Sie zusätzliches Papier bitte nur an, falls Sie den gesamten Platz aufgebraucht haben.
- Die Tackernadel darf nicht gelöst werden.
- Als Hilfsmittel ist ein beschriebenes A4-Papier erlaubt.
- Einlesezeit: 15 min
  Bearbeitungszeit: 2 h

	Mögliche Punkte					Erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
Aufg. 1	2	3	4	–	9				–	
Aufg. 2	1	3	2	4	10					
Aufg. 3	1	1	1	7	10					
Aufg. 4	2	3	2	–	7				–	
Aufg. 5	3	3	–	–	6			–	–	
Aufg. 6	2	5	3	–	10				–	
Aufg. 7	1	3	1	3	8					
Σ					60					

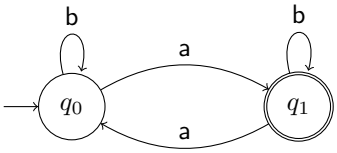
Problem 1: Endliche Automaten $2 + 3 + 4 = 9$ Punkte

Sei $\mathcal{A}$ ein endlicher Automat über dem Alphabet $\Sigma = \{0, 1\}$ und sei $\mathcal{A}_{\text{ins}}$ ein endlicher Automat über dem Alphabet $\Sigma_{\text{ins}} = \{\mathsf{a}, \mathsf{b}\}$. Diese sind durch folgende Übergangsgraphen (mit implizitem Fehlerzustand) definiert:

$\mathcal{A}$:



$\mathcal{A}_{\text{ins}}$:



Sei L die Sprache, die von $\mathcal{A}$ erkannt wird, und L_{ins} die Sprache, die von $\mathcal{A}_{\text{ins}}$ erkannt wird.

(a) Geben Sie L und L_{ins} an.

Wir betrachten nun ein Wort $w \in L$ und fügen nach jeder 1 in w ein beliebiges Wort aus L_{ins} ein.

Beispiel: $w = 011010 \rightsquigarrow 01\mathsf{a}1\mathsf{a}01\mathsf{babbaa}0, 01\mathsf{abb}1\mathsf{a}01\mathsf{bab}0$

Die Sprache aller Wörter, die so entstehen können, bezeichnen wir mit $\alpha(w, 1, L_{\text{ins}})$. Außerdem definieren wir

$$\alpha(L, 1, L_{\text{ins}}) = \bigcup_{w \in L} \alpha(w, 1, L_{\text{ins}}).$$

Das heißt, $\alpha(L, 1, L_{\text{ins}})$ enthält alle Wörter, die entstehen, wenn wir mit einem beliebigen Wort aus L beginnen und nach jeder 1 ein beliebiges Wort aus L_{ins} einfügen.

(b) Geben Sie einen nichtdeterministischen endlichen Automaten für $\alpha(L, 1, L_{\text{ins}})$ an, mit L und L_{ins} wie oben.

(c) Seien nun L und L_{ins} beliebige reguläre Sprachen über $\Sigma = \{\mathsf{0}, \mathsf{1}\}$ bzw. $\Sigma_{\text{ins}} = \{\mathsf{a}, \mathsf{b}\}$. Zeigen Sie, dass $\alpha(L, \mathsf{1}, L_{\text{ins}})$ ebenfalls regulär ist. Zeigen Sie dazu, dass ein Automat $\mathcal{A}_\alpha$ existiert, der $\alpha(L, \mathsf{1}, L_{\text{ins}})$ erkennt.

Problem 2: Grammatiken/Kodierungen $1 + 3 + 2 + 4 = 10$ Punkte

In dieser Aufgabe zeigen wir, dass neben dem Startsymbol zwei Variablen ausreichen, um beliebige Grammatiken zu konstruieren. Wir betrachten zunächst folgende Beispielgrammatik.

Beispiel: $G_{\text{Bsp}} = (\Sigma_{\text{Bsp}}, V_{\text{Bsp}}, S_{\text{Bsp}}, R_{\text{Bsp}})$ mit $\Sigma_{\text{Bsp}} = \{2, 3, 4\}$, $V_{\text{Bsp}} = \{S_{\text{Bsp}}, X_1, X_2, X_3, X_4\}$ und

$$\begin{aligned} R_{\text{Bsp}} = \{ \quad & S_{\text{Bsp}} \to X_2 \\ & X_1 \to X_2 \\ & X_2 \to 2 \\ & X_3 \to 3 \\ & X_4 \to 4 \\ & X_1X_1 \to X_1X_1X_2X_3 \\ \}. & \end{aligned}$$

(a) Welche Sprache erzeugt G_{Bsp}?

Sei Σ ein Alphabet und $G = (\Sigma, V, S, R)$ eine Grammatik mit der Variablenmenge $V = \{S, X_1, \ldots, X_n\}$ für ein $n \in \mathbb{N}$. Für eine Funktion $f\colon \{X_1, \ldots, X_n\} \to \{A, B\}^*$ definieren wir die *f-Kodierung von G* als die Grammatik $G' = (\Sigma, V', S, R')$ mit $V' = \{S, A, B\}$ und der Regelmenge R', die aus R entsteht, indem jedes X_i für $i \in \{1, \ldots, n\}$ durch $f(X_i)$ ersetzt wird.

Beispiel: Sei f gegeben durch $f(X_1) = ABA$, $f(X_2) = ABBA$, $f(X_3) = ABBBA$ und $f(X_4) = ABBBBA$. Dann ergibt sich die folgende Regelmenge R' für die f-Kodierung G'_{Bsp} von G_{Bsp}:

$$\begin{aligned} R' = \{ \quad & S_{\text{Bsp}} \to ABBA \\ & ABA \to ABBA \\ & ABBA \to 2 \\ & ABBBA \to 3 \\ & ABBBBA \to 4 \\ & ABA\,ABA \to ABA\,ABA\,ABBA\,ABBBA \\ \}. & \end{aligned}$$

Dr. Meta möchte die Variablen möglichst effizient kodieren und schlägt vor, eine Huffman-Kodierung zu verwenden. Für eine Grammatik $G = (\Sigma, V, S, R)$ mit $V = \{S, X_1, \ldots, X_n\}$ definieren wir die Huffman-Kodierung von G wie folgt. Für ein X_i mit $i \in \{1, \ldots, n\}$ ist die Wahrscheinlichkeit p_i definiert als die relative Häufigkeit von X_i in R, das heißt

$$p_i = \frac{\text{Anzahl Vorkommen von } X_i \text{ in } R}{\sum_{j=1}^{n} \text{Anzahl Vorkommen von } X_j \text{ in } R}$$

Dabei werden die Vorkommen auf *beiden* Seiten der Regeln gezählt.

Beispiel: Die Anzahl Vorkommen von X_1 in R ist 5, d.h. $p_1 = 5/12$.

Sei f eine Huffman-Kodierung für $\{X_1, \ldots, X_n\}$ mit den so definierten Wahrscheinlichkeiten $p_1, \ldots, p_n$. Die f-Kodierung von G heißt dann auch *Huffman-Kodierung von G*. Der zu f gehörende Huffman-Baum heißt *Huffman-Baum der Grammatik G*.

(b) Geben Sie einen Huffman-Baum für die Grammatik G_{Bsp} an. Sortieren Sie dafür die Variablen von links nach rechts nach absteigender Wahrscheinlichkeit und wählen Sie A für den linken Ast und B für den rechten Ast. Geben Sie die Kodierung zu Ihrem Huffman-Baum durch Wörter aus $\{A, B\}^*$ in folgender Tabelle an.

Variable	Kodierung
X_1	
X_2	
X_3	
X_4	

(c) Zeigen Sie, dass Dr. Metas Ansatz im Allgemeinen nicht funktioniert. Zeigen Sie hierfür, dass die Grammatik G'_{Bsp}, die durch die Huffman-Kodierung entsteht, nicht dieselbe Sprache wie G_{Bsp} erzeugt.

(d) Wir möchten dennoch zeigen, dass neben dem Startsymbol zwei Variablen ausreichen. Sei $G = (\Sigma, V, S, R)$ eine beliebige Grammatik mit $V = \{S, X_1, \ldots, X_n\}$ für ein $n \in \mathbb{N}$. Zeigen Sie, dass eine äquivalente Grammatik $G' = \{\Sigma, V', S, R'\}$ mit $V' = \{S, A, B\}$ existiert.

Hinweis: Verwenden Sie eine Unär-Kodierung mit Trennzeichen, das heißt $f(X_i) = AB^iA$ für $i \in \{1, \ldots, n\}$, siehe Beispiel.

Problem 3: NP-Vollständigkeit $1 + 1 + 1 + 7 = 10$ Punkte

Aus der Vorlesung kennen Sie das folgende NP-vollständige Problem PARTITION:

PARTITION	
Gegeben:	Eine endliche Menge M, eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$
Frage:	Existieren disjunkte Teilmengen $M_1, M_2 \subseteq M$ mit $M_1 \cup M_2 = M$ und $w(M_1) = w(M_2)$?

Für eine Menge M' definieren wir das Gewicht von M' als die Summe der Gewichte der Elemente von M', d.h. $w(M') \coloneqq \sum_{m \in M'} w(m)$.

Wir betrachten nun das folgende Problem PARTITION-IN-3:

PARTITION-IN-3	
Gegeben:	Eine endliche Menge M, eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$
Frage:	Existieren paarweise disjunkte Teilmengen $M_1, M_2, M_3 \subseteq M$ mit $M_1 \cup M_2 \cup M_3 = M$ und $w(M_1) = w(M_2) = w(M_3)$?

(a) Geben Sie für folgende Instanz von PARTITION eine Lösung an.

$$\begin{aligned} M &= \{a, b, c, d, e, f\} \\ w(a) &= 0 \\ w(b) &= 2 \\ w(c) &= 3 \\ w(d) &= 4 \\ w(e) &= 5 \\ w(f) &= 6 \end{aligned}$$

(b) Sei (M, w) die PARTITION-Instanz aus (a). Konstruieren Sie eine Ja-Instanz (M', w') für PARTITION-IN-3 mit $M' = M \cup \{g\}$ und $w'(m) = w(m)$ für alle $m \in M$. Es genügt, wenn Sie das Gewicht von g angeben.

(c) Zeigen Sie, dass PARTITION-IN-3 in NP liegt. Geben Sie an, woraus ein Lösungsvorschlag für eine PARTITION-IN-3-Instanz besteht. Geben Sie außerdem die Laufzeit der Überprüfung im $\mathcal{O}$-Kalkül an und begründen Sie diese.

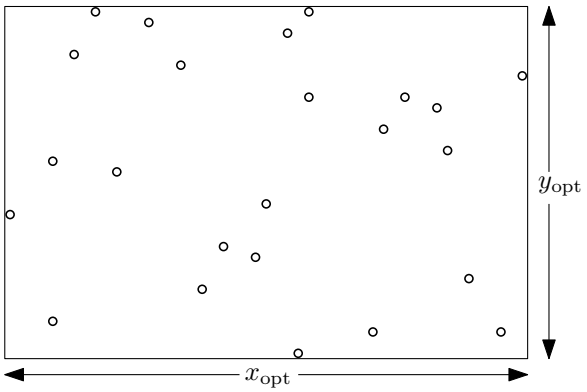
(d) Zeigen Sie, dass PARTITION-IN-3 NP-schwer ist.

Problem 4: Approximation $2 + 3 + 2 = 7$ Punkte

Beim Minimierungsproblem SMALLESTBOUNDINGRECTANGLE ist eine Menge $P \subset \mathbb{R}^2$ von Punkten in der Ebene gegeben. Gesucht ist ein achsenparalleles[1] Rechteck mit minimalem Flächeninhalt, das alle Punkte in P enthält. Wir bezeichnen die Seitenlängen einer optimalen Lösung mit x_{opt} und y_{opt}.

[1]D.h. die Seiten des Rechtecks sind parallel zur x- bzw. y-Achse.

Beispiel:



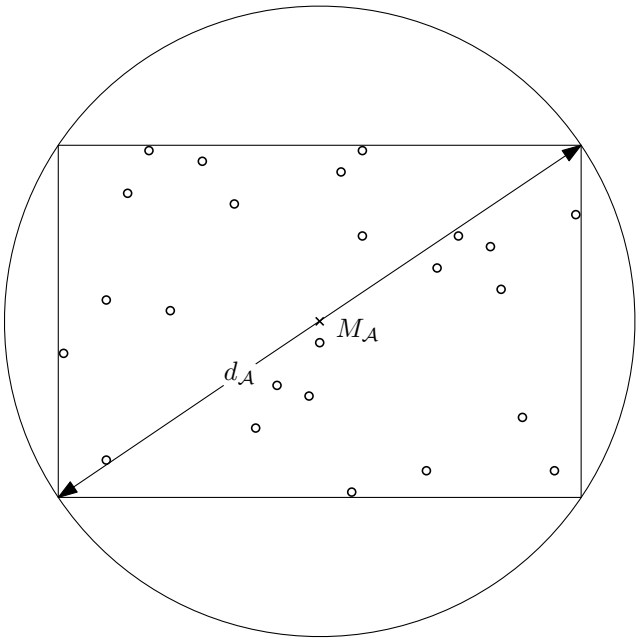
(a) Zeigen Sie, dass SMALLESTBOUNDINGRECTANGLE optimal in Linearzeit gelöst werden kann.

Wir betrachten nun das Minimierungsproblem SMALLESTBOUNDINGCIRCLE. Wieder ist eine Menge $P \subset \mathbb{R}^2$ von Punkten in der Ebene gegeben. Gesucht ist ein Kreis mit minimalem Durchmesser d_{opt}, der alle Punkte in P enthält.

(b) Zeigen Sie: $d_{\text{opt}} \geqslant \max(x_{\text{opt}}, y_{\text{opt}})$.

Wir betrachten nun folgenden Algorithmus $\mathcal{A}$ für SMALLESTBOUNDINGCIRCLE: Berechne zunächst eine optimale Lösung für SMALLESTBOUNDINGRECTANGLE. Seien $M_{\mathcal{A}}$ und $d_{\mathcal{A}}$ der Mittelpunkt und die Diagonallänge des berechneten Rechtecks. Erzeuge dann einen Kreis mit Mittelpunkt $M_{\mathcal{A}}$ und Durchmesser $d_{\mathcal{A}}$. Sie können davon ausgehen, dass es möglich ist, in Polynomialzeit Wurzeln zu ziehen.

Beispiel:



(c) Zeigen Sie, dass $\mathcal{A}$ ein polynomieller Approximationsalgorithmus für SMALLESTBOUNDINGCIRCLE mit relativer Gütegarantie $\sqrt{2}$ ist.

Problem 5: Kellerautomaten 3 + 3 = 6 Punkte

(a) Seien Σ ein beliebiges Alphabet, $a, b \in \Sigma$ zwei beliebige Symbole aus dem Alphabet, und $k \in \mathbb{N}$ eine Konstante. Beschreiben Sie, wie ein deterministischer Kellerautomat die Sprache $L(a, b, k) = \{a^n b^{kn} \mid n \in \mathbb{N}\}$ erkennen kann.

Wir betrachten nun deterministische Kellerautomaten, die ihren Lesekopf frei auf dem Eingabeband bewegen können. Wir sagen, dass die Ausführung eines solchen Kellerautomaten beendet ist, sobald er das Feld direkt hinter dem letzten Zeichen der Eingabe erreicht. Das Akzeptanzverhalten ist dann wie üblich über einen leeren Stack oder einen akzeptierenden Endzustand definiert.

(b) Beschreiben Sie, wie ein solcher Kellerautomat die Sprache $L = \{\mathtt{0}^n\mathtt{1}^{2n}\mathtt{2}^{6n} \mid n \in \mathbb{N}\}$ erkennen kann.

Hinweis: Sie dürfen den Kellerautomaten aus Teilaufgabe (a) als Subroutine verwenden, auch wenn Sie Teilaufgabe (a) nicht gelöst haben. Falls Sie den Kellerautomaten anpassen müssen, gehen Sie auf Unterschiede ein.

Problem 6: Falltür-Turingmaschine $2 + 5 + 3 = 10$ Punkte

(a) Beschreiben Sie, wie die Sprache $L = \{1^k0^m1^m0^n \mid k, m, n \in \mathbb{N}_0\}$ von einer deterministischen Turingmaschine entschieden werden kann. Schreiben Sie dafür nur auf Feldern der Eingabe.

Sei $F \subseteq \Sigma^*$ eine Sprache über einem Alphabet Σ. Eine *F-Falltür-Turingmaschine* (F-FTTM) ist eine deterministische Turingmaschine, die zusätzlich über ein *Falltür-Modul* verfügt. Nach jeder Anwendung der Überführungsfunktion wird das Falltür-Modul aktiv und macht folgende Überprüfung: Falls die F-FTTM im letzten Schritt das Feld verändert hat (d.h. ein anderes Zeichen geschrieben hat als gelesen wurde), wird überprüft, ob das Wort auf dem Band in F ist. Falls ja, so wird die „Falltür“ ausgelöst, d.h. der gesamte Bandinhalt wird gelöscht.

Bei der Überprüfung werden alle Zeichen auf dem Band, die nicht in Σ enthalten sind, ignoriert. Das heißt, wenn $\Sigma = \{0, 1\}$ und $00 \in F$, so wird das Band auch dann gelöscht, wenn $0x0$ mit $x \notin \Sigma$ auf dem Band steht. Das Falltür-Modul arbeitet separat von der endlichen Kontrolleinheit, d.h. Zustand und Kopfposition werden durch das Falltür-Modul nicht verändert. Falls der Bandinhalt nicht gelöscht wird, so wird er durch das Falltür-Modul nicht verändert.

(b) Sei $\Sigma = \{0, 1\}$ und $F = \{w \in \Sigma^* \mid w = w^R\}$ die Palindromsprache. Beschreiben Sie, wie eine F-Falltür-Turingmaschine die Sprache $L = \{1^k 0^m 1^m 0^n \mid k, m, n \in \mathbb{N}_0\}$ entscheidet.

Hinweis: Verhindern Sie, dass die Falltür ausgelöst wird, indem Sie die Felder vor und nach der Eingabe nutzen. Sie dürfen die Turingmaschine aus Aufgabenteil (a) als Subroutine nutzen, auch wenn Sie (a) nicht bearbeitet haben. Falls Sie die Turingmaschine anpassen müssen, gehen Sie auf Unterschiede ein.

(c) Zeigen Sie, dass es Sprachen F und L gibt, sodass L von einer F-FTTM entschieden werden kann, aber von keiner klassischen Turingmaschine.

Problem 7: Entscheidbarkeit $1 + 3 + 1 + 3 = 8$ Punkte

Sei Σ ein beliebiges, aber festes Alphabet. Betrachten Sie das folgende Entscheidungsproblem Π: Gegeben kontextfreie Grammatiken G_1 und G_2 über Σ, ist der Schnitt $L(G_1) \cap L(G_2)$ leer? In der Vorlesung haben Sie gesehen, dass dieses Problem unentscheidbar ist.

(a) Beschreiben Sie eine Turingmaschine $\mathcal{M}'$, die als Eingabe eine kontextfreie Grammatik G und ein Wort $w \in \Sigma^*$ bekommt und entscheidet, ob $w \in L(G)$ gilt.

(b) Zeigen Sie, dass das Komplement von Π semi-entscheidbar ist.

(c) Ist Π auch semi-entscheidbar? Beweisen Sie.

(d) Seien nun statt kontextfreien Grammatiken zwei Turingmaschinen gegeben. Wir erhalten das neue Entscheidungsproblem Π': Gegeben zwei Turingmaschinen $\mathcal{M}_1$ und $\mathcal{M}_2$ mit Eingabealphabet Σ, ist der Schnitt von $L(\mathcal{M}_1) \cap L(\mathcal{M}_2)$ leer? Zeigen Sie, dass das Komplement von Π' ebenfalls semi-entscheidbar ist.

Hinweis: Beachten Sie, dass die Turingmaschinen $\mathcal{M}_1$ bzw. $\mathcal{M}_2$ die Sprachen $L(\mathcal{M}_1)$ bzw. $L(\mathcal{M}_2)$ nicht entscheiden müssen.