2. Klausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2019/2020

Hier Aufkleber mit Name und Matrikelnummer anbringen

Vorname: _____

Nachname: _____

Matrikelnummer: _____

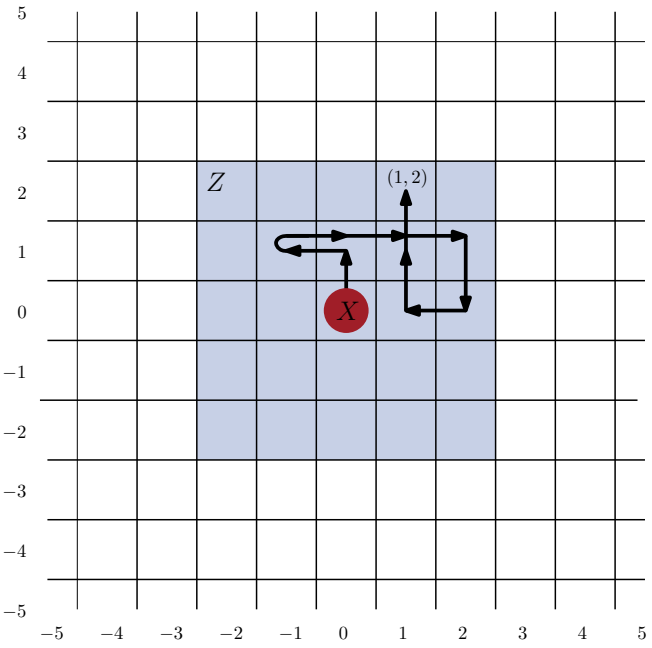
Beachten Sie:

- 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.
- Es sind keine Hilfsmittel zugelassen.

	Mögliche Punkte							Erreichte Punkte						
	a	b	c	d	e	f	Σ	a	b	c	d	e	f	Σ
Aufg. 1	3	4	1	–	–	–	8				–	–	–	
Aufg. 2	1	2	2	3	–	–	8					–	–	
Aufg. 3	3	4	–	–	–	–	7			–	–	–	–	
Aufg. 4	1	3	1	2	1	2	10							
Aufg. 5	8						8							
Aufg. 6	2	1	3	1	2	3	12							
Aufg. 7	1	1	2	2	1	–	7						–	
Σ							60							

Problem 1: Reguläre Sprachen $3 + 4 + 1 = 8$ Punkte

Ein Roboter X bewegt sich auf einem unendlichen zweidimensionalen Gitter. Seine Startposition ist das Feld $(0, 0)$. In jedem Schritt kann sich der Roboter ein Feld nach oben, unten, links oder rechts bewegen.



Wir kodieren eine Folge von Zügen als Wort über dem Alphabet $\Sigma = \{\texttt{O}, \texttt{U}, \texttt{L}, \texttt{R}\}$. Zum Beispiel kodiert das Wort `OLRRRULOO` die Zugfolge oben-links-rechts-rechts-rechts-unten-links-oben-oben, die in der Abbildung dargestellt ist. Nach dieser Zugfolge befindet sich der Roboter auf dem Feld $(1, 2)$.

Wir nennen das Quadrat von $5{\times}5$ Feldern um den Startpunkt $(0, 0)$ herum das *Zuhause* des Roboters. Formal ist das die Menge von Feldern $Z = \{(i, j) \mid i, j \in \{-2, -1, 0, 1, 2\}\}$.

Zeigen Sie:

(a) Die Sprache $L_1 = \{w \in \Sigma^* \mid X \text{ verlässt während Zugfolge } w \text{ nie } Z\}$ ist regulär. Geben Sie dazu einen endlichen Automaten an, der L_1 akzeptiert.

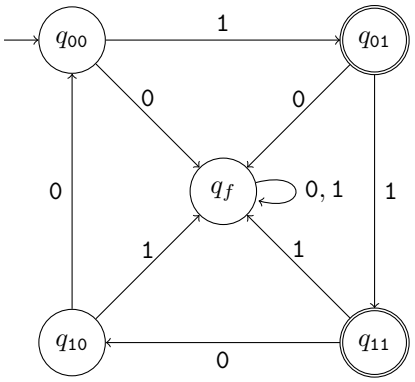
(b) Die Sprache $L_2 = \{w \in \Sigma^* \mid X \text{ befindet sich am Ende von } w \text{ in } Z\}$ ist nicht regulär. Verwenden Sie dazu das Pumping-Lemma.

(c) Die Sprache $L_3 = \{w \in \Sigma^* \mid X \text{ befindet sich am Ende von } w \text{ nicht in } Z\}$ ist nicht regulär.

Hinweis: Sie müssen hierzu nicht das Pumping-Lemma verwenden!

Problem 2: Endliche Automaten mit Gedächtnis $1 + 2 + 2 + 3 = 8$ Punkte

Gegeben sei folgender deterministischer endlicher Automat $\mathcal{A}$:

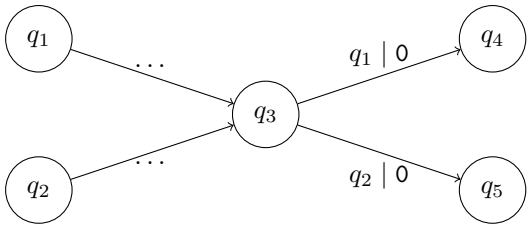


(a) Geben Sie einen regulären Ausdruck für die Sprache $L(\mathcal{A})$ an, die von $\mathcal{A}$ erkannt wird.

(b) Zeigen Sie, dass $\mathcal{A}$ minimal ist.

Ein *deterministischer endlicher Automat mit Gedächtnis* (DEAG) ist ein deterministischer endlicher Automat mit einer Überführungsfunktion $\delta\colon Q \times Q \times \Sigma \to Q$, die als Eingabe nicht nur den aktuellen Zustand bekommt, sondern auch den Zustand, der im vorigen Schritt besucht wurde.

Beispiel:



Wenn im Zustand q_3 eine 0 gelesen wird, hängt es vom vorigen Zustand ab, welcher Übergang genommen wird: Wenn q_3 über q_1 erreicht wurde, wird der Übergang nach q_4 genommen. Wenn q_3 dagegen über q_2 erreicht wurde, wird der Übergang nach q_5 genommen.

Achtung: Wenn gerade das erste Zeichen der Eingabe gelesen wird, befindet sich der Automat im Startzustand s und es gab noch keinen vorigen Schritt. In diesem Fall legen wir fest, dass der Automat im „vorigen Schritt“ ebenfalls im Startzustand war.

(c) Geben Sie einen DEAG mit 2 Zuständen (ohne Fehlerzustand) an, der $L(\mathcal{A})$ erkennt.

(d) Beschreiben Sie, wie für jeden DEAG ein äquivalenter DEA (ohne Gedächtnis) konstruiert werden kann. Begründen Sie, warum Ihre Konstruktion korrekt ist!

Problem 3: Turing-Maschinen mit Lesezeichen $3 + 4 = 7$ Punkte

Wir erweitern das Berechnungsmodell der Turing-Maschine (ein Kopf, ein Band, eine Spur) folgendermaßen. Zusätzlich zum Kopf verfügt die Turing-Maschine über ein *Lesezeichen*. Anfangs befindet sich dieses auf der Startposition des Kopfes. Neben den üblichen Operationen kann sich die Turing-Maschine nun in jedem Schritt entschließen, das Lesezeichen auf die aktuelle Kopfposition zu verschieben. Das Bandsymbol an dieser Position wird dadurch nicht gelöscht.

Außerdem ist es möglich, von der aktuellen Position zum Lesezeichen zu springen. Sowohl das Verschieben des Lesezeichens als auch das Springen zum Lesezeichen benötigen jeweils einen Zeitschritt. Durch das Springen zum Lesezeichen können also größere Distanzen in einem Zeitschritt überwunden werden. Wir bezeichnen dieses Modell als *Lesezeichen-Turing-Maschine* (LTM).

Betrachten Sie folgende (herkömmliche) Turing-Maschine, die die Palindromsprache erkennt. Sie startet auf dem linkesten Eingabesymbol, merkt sich dieses, löscht es, geht zum rechtesten Symbol und vergleicht dieses mit dem gemerkten Symbol. Sind die Symbole unterschiedlich, wird das Wort abgelehnt. Andernfalls wird das rechteste Symbol gelöscht, die Turing-Maschine geht zurück zum linkesten verbleibenden Symbol und der Ablauf beginnt erneut. Die Eingabe wird akzeptiert, wenn alle Symbole der Eingabe gelöscht wurden.

(a) Beschreiben Sie, wie eine LTM auf Grundlage der oben beschriebenen TM das Lesezeichen ausnutzen kann, um die Palindromsprache ungefähr doppelt so schnell zu entscheiden.

(b) Zeigen Sie, dass eine LTM eine Sprache höchstens doppelt so schnell wie eine herkömmliche Turing-Maschine entscheiden kann.

Problem 4: Entscheidbarkeit $1 + 3 + 1 + 2 + 1 + 2 = 10$ Punkte

Sei $L \neq \emptyset$ eine entscheidbare Sprache. Betrachte das Suchproblem Π:

Gegeben:	Turing-Maschine $\mathcal{M}$, für die gilt: • Es gibt ein Wort $x \in L$, sodass $L(\mathcal{M}) = L \setminus \{x\}$ gilt.
Ausgabe:	x

(a) Zeigen Sie, dass $L(\mathcal{M})$ entscheidbar ist.

Anmerkung: $\mathcal{M}$ hält nicht notwendigerweise auf allen Eingaben.

(b) Gehen Sie ab jetzt davon aus, dass M auf allen Eingaben hält. Zeigen Sie, dass eine Turing-Maschine $\mathcal{M}_\Pi$ existiert, die Π löst, d.h. für eine Eingabe $\mathcal{M}$ das gesuchte x berechnet.

Nun erlauben wir zusätzlich auch den Fall, dass $L(\mathcal{M}) = L$ gilt, also dass kein Wort x fehlt. In diesem Fall soll das Platzhalterzeichen $\perp$ ausgegeben werden. Wir erhalten also das Suchproblem Λ:

Gegeben:	Turing-Maschine $\mathcal{M}$, für die gilt: • **Fall 1:** Es gibt ein Wort $x \in L$, sodass $L(\mathcal{M}) = L \setminus \{x\}$ gilt. **oder** • **Fall 2:** $L(\mathcal{M}) = L$
Ausgabe:	• **In Fall 1:** x • **In Fall 2:** $\perp$

(c) Wieso kann $\mathcal{M}_\Pi$ nicht angepasst werden, um Λ zu lösen?

Nun interessiert uns das konkrete x nicht mehr. Wir wollen nur noch wissen, ob $L(\mathcal{M}) = L$ gilt. Wir erhalten das Entscheidungsproblem Γ:

Gegeben:	Turing-Maschine $\mathcal{M}$, für die gilt: • **Fall 1:** Es gibt ein Wort $x \in L$, sodass $L(\mathcal{M}) = L \setminus \{x\}$ gilt. **oder** • **Fall 2:** $L(\mathcal{M}) = L$
Frage:	Gilt Fall 1 oder Fall 2?

Wir wollen nun zeigen, dass Γ unentscheidbar ist. Seien zunächst ein Wort $x \in L$, eine Turing-Maschine $\mathcal{N}$ und ein Wort $v \in \Sigma^*$ gegeben. Wir wollen eine Turing-Maschine $\mathcal{X}$ konstruieren, für die gilt:

$$L(\mathcal{X}) = \begin{cases} L & \text{falls } \mathcal{N} \text{ auf } v \text{ hält} \\ L \setminus \{x\} & \text{sonst} \end{cases}$$

(d) Vervollständigen Sie folgende Aussage:

$$w \in L(\mathcal{X}) \Leftrightarrow \begin{cases} \qquad\qquad & \text{falls } w \neq x \\ \qquad\qquad & \text{falls } w = x \end{cases}$$

(e) Beschreiben Sie, wie $\mathcal{X}$ konstruiert werden kann.

(f) Zeigen Sie, dass Γ unentscheidbar ist. Verwenden Sie nicht den Satz von Rice! Nehmen Sie an, dass eine Turing-Maschine $\mathcal{M}_\Gamma$ existiert, die Γ entscheidet. Führen Sie diese Annahme zum Widerspruch, indem Sie eine Turing-Maschine $\mathcal{M}_\mathcal{H}$ konstruieren, die das Halteproblem entscheidet.

Setzen Sie die Turing-Maschine $\mathcal{X}$ auf geeignete Weise ein. Sie können ohne Beweis die Tatsache verwenden, dass sich ein Wort $x \in L$ in endlicher Zeit generieren lässt.

Problem 5: NP-Vollständigkeit 8 Punkte

Aus der Vorlesung kennen Sie das NP-vollständige Entscheidungsproblem SAT.

Eine Klausel heißt *monoton*, wenn sie entweder nur positive Literale oder nur negative Literale enthält. Zum Beispiel sind $x \vee y \vee z$ und $\bar{x} \vee \bar{y}$ monoton, wohingegen $x \vee \bar{y} \vee z$ nicht monoton ist. Eine SAT-Instanz heißt *monoton*, wenn alle darin vorkommenden Klauseln monoton sind. Es ergibt sich das Entscheidungsproblem MONOTONESAT (MSAT):

Gegeben:	Menge U von n Variablen Menge C von k monotonen Klauseln über U
Frage:	Existiert eine Wahrheitsbelegung von U, die C erfüllt?

Zeigen Sie, dass MONOTONESAT NP-vollständig ist. Geben Sie bei Ihrer Reduktion explizit an, von welchem Problem auf welches reduziert wird!

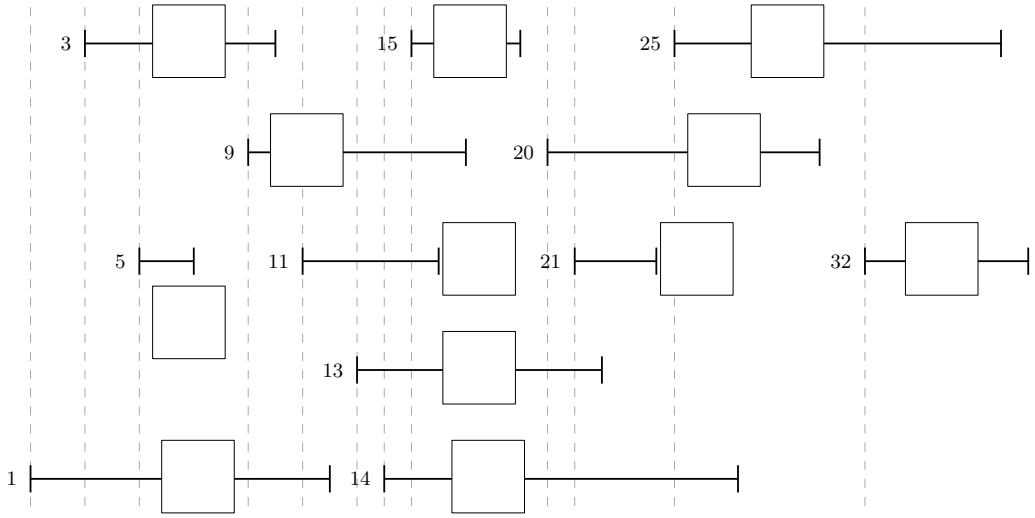
Problem 6: Approximation $2 + 1 + 3 + 1 + 2 + 3 = 12$ Punkte

Gegeben sei eine Menge M von n endlichen Intervallen über den natürlichen Zahlen. Sie dürfen davon ausgehen, dass alle Intervallgrenzen und -größen paarweise verschieden sind. Eine Färbung von M ist eine Funktion $\varphi : M \to C$, sodass für $x, y \in M$ mit $x \neq y$ und $x \cap y \neq \emptyset$ gilt, dass $\varphi(x) \neq \varphi(y)$. Überlappende Intervalle müssen also verschiedene Farben haben. Eine k-Färbung von M ist eine Färbung $\varphi : M \to \{1, 2, \ldots, k\}$. Das Optimierungsproblem IntervalleFärben (IF) besteht darin, das kleinste k zu bestimmen, sodass es eine k-Färbung von M gibt.

Algorithmus A funktioniert wie folgt. Wähle die natürlichen Zahlen $\{1, 2, \ldots\}$ als Farbkandidaten. Betrachte die Intervalle **gemäß ihrer unteren Grenze in aufsteigender Reihenfolge**. Berechne bei Betrachtung eines Intervalls $x = [a, b]$ alle Intervalle außer x, welche a enthalten. (Das müssen nicht alle Intervalle sein, die x schneiden.) Wähle die kleinste natürliche Zahl, die keinem dieser Intervalle zugewiesen wurde, als Farbe von x.

(a) Führen Sie Algorithmus A auf folgendem Beispiel aus.

Schreiben Sie die Farben in die weißen Quadrate. Die Zahlen entsprechen den unteren Grenzen der jeweiligen Intervalle.



(b) Beweisen Sie, dass Algorithmus A eine Färbung von M berechnet.

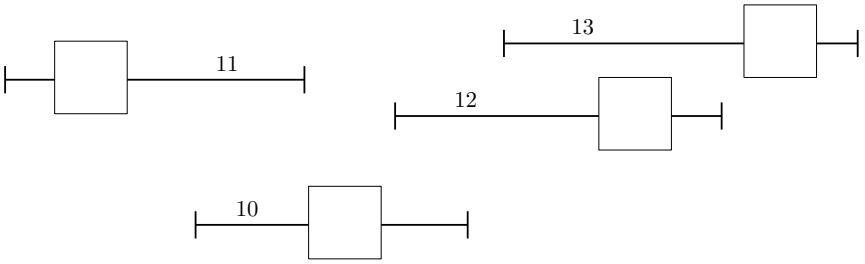
(c) Beweisen Sie, dass Algorithmus A eine Färbung minimaler Größe berechnet.

Hinweis: Betrachten Sie beim Berechnen einer k-Färbung den Zeitpunkt, in dem die k-te Farbe erstmals zugewiesen wird.

Algorithmus B funktioniert wie folgt. Wähle wieder die natürlichen Zahlen $\{1, 2, \dots\}$ als Farbkandidaten. Betrachte die Intervalle **gemäß ihrer Größe in absteigender Reihenfolge**. Berechne bei Betrachtung eines Intervalls $x = [a, b]$ alle Intervalle außer x, die a oder b enthalten. (Das müssen nicht alle Intervalle sein, die x schneiden.) Wähle die kleinste natürliche Zahl, die keinem dieser Intervalle zugewiesen wurde als Farbe von x.

(d) Führen Sie Algorithmus B auf folgendem Beispiel aus.

Schreiben Sie die Farben in die weißen Quadrate. Die Zahlen entsprechen den Größen der jeweiligen Intervalle.



(e) Zeigen Sie, dass Algorithmus B eine Färbung von M berechnet.

(f) Beweisen Sie, dass Algorithmus B ein Approximationsalgorithmus mit relativer Gütegarantie 2 für das Optimierungsproblem IntervalleFärben ist.

Problem 7: Grammatiken $1 + 1 + 2 + 2 + 1 = 7$ Punkte

Sei $G = (\Sigma, V, S, R)$ eine Grammatik mit endlichem Alphabet Σ, Variablenmenge V, Startsymbol $S \in V$ und einer Menge von Ableitungsregeln R. Jede Ableitungsregel ist von einem der folgenden drei Typen:

(1) $S \to \varepsilon$

(2) $A \to a$ mit $A \in V$ und $a \in \Sigma$

(3) $\alpha \to \beta$ mit $\alpha \in V^\star$, $\beta \in (V \setminus \{S\})^\star$, wobei $1 \leq |\alpha| \leq 2$ und $|\alpha| \leq |\beta|$

Insbesondere ist G eine kontextsensitive Grammatik.

(a) Zeigen Sie, dass es eine zu G äquivalente kontextsensitive Grammatik gibt, bei der für alle Regeln vom Typ (3) zusätzlich $|\beta| \leq 2$ gilt.

(b) Gibt es eine zu G äquivalente Grammatik, bei der $|\beta| \leq 1$ für alle Regeln vom Typ (3) gilt? Beweisen oder widerlegen Sie!

Ab jetzt lassen wir die Forderung $|\alpha| \leq 2$ fallen. Die Ableitungsregeln haben also die Form:

(1) $S \to \varepsilon$

(2) $A \to a$ mit $A \in V$ und $a \in \Sigma$

(3) $\alpha \to \beta$ mit $\alpha \in V^\star$, $\beta \in (V \setminus \{S\})^\star$ und $1 \leq |\alpha| \leq |\beta|$

Das sind genau die Forderungen, die wir an kontextsensitive Grammatiken stellen.

(c) Betrachten Sie den Fall, dass unter Ausnahme genau einer Regel $ABC \to XYZ$ für jede Regel von Typ (3) $|\alpha| = |\beta| = 2$ gilt. Konstruieren Sie eine zu G äquivalente Grammatik, bei der für jede Regel (ohne Ausnahme) von Typ (3) $|\alpha| = |\beta| = 2$ gilt.

(d) Erweitern Sie Ihren Ansatz aus (c) um zu zeigen, dass Sie für Ableitungen von Typ (3) $|\alpha| \leq 2$ fordern können.

(e) Gibt es eine zu G äquivalente Grammatik, bei der $|\alpha| \leq 1$ für alle Regeln vom Typ (3) gilt? Beweisen oder widerlegen Sie!