Einführung in die Theoretische Informatik
Klausur, Nebentermin SoSe 2021, 21. September 2021

Aufgabe 1: Informationstheorie (12 Punkte)

(a) Quelle konstruieren (6 Punkte)

Entscheiden Sie für die folgenden Alphabete Σ, ob eine Quelle (Σ, p) existiert, sodass jeder endliche Informationsgewinn $\mathcal{I}(p_\sigma), \sigma \in \Sigma$, ganzzahlig ist.

(i) $\Sigma = \{a\}$, **(ii)** $\Sigma = \{a, b\}$, **(iii)** $\Sigma = \{a, b, c\}$.

Falls ja, geben Sie die Wahrscheinlichkeiten und Informationsgewinne der jeweiligen Symbole an. Falls nein, begründen Sie kurz.

(b) Präfix-Code (6 Punkte)

Konstruieren Sie einen Huffman-Code $\mathbb{C}$ für die in der Tabelle angegebene Quelle (Σ, p). Geben Sie sowohl den finalen Baum als auch die Codewörter der einzelnen Symbole an.

σ	a	b	c	d	e
p_σ	$\frac{3}{10}$	$\frac{1}{10}$	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{3}{10}$

Aufgabe 2: Umwandlung RegEx → NDEA (8 Punkte)

Wandeln Sie den folgenden regulären Ausdruck – gemäß dem Vorgehen aus der Vorlesung – in einen nichtdeterministischen endlichen Automaten um.

$$a^+(b \mid ba)^*$$

Aufgabe 3: Pumping Lemma (10 Punkte)

(a) Zerlegung (2 Punkte)

Sei L eine reguläre Sprache. Nehmen Sie an, dass es ein Wort $w \in L$ gibt, sodass für w keine Zerlegung gemäß des Pumping Lemmas existiert.

Warum ist das kein Widerspruch zur Regularität von L?

(b) Anwendung (8 Punkte)

Sei $L \coloneqq \{ba^i b^j a^k \mid i = (j{-}2){\cdot}(k{+}2),\ j \geq 2,\ k \geq 2\}$.

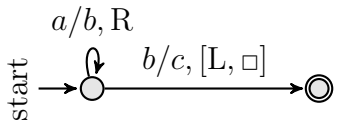
Zeigen Sie mithilfe des Pumping Lemmas, dass L nicht regulär ist.

Aufgabe 4: Turingmaschinen (10 Punkte)

Wir **erweitern** die TM-Notation aus der Vorlesung um eine neue Beschriftungsmöglichkeit für Übergänge. Seien $\sigma_1, \sigma_2, \sigma_3 \in \Gamma$ und $\mathrm{D} \in \{\mathrm{L}, \mathrm{R}\}$. Das Verhalten einer TM bei einem Übergang mit Beschriftung „$\sigma_1/\sigma_2, [\mathrm{D}, \sigma_3]$“ wird folgendermaßen definiert (wobei die ersten beiden Punkte identisch zum Verhalten eines Übergangs mit Beschriftung „$\sigma_1/\sigma_2, \mathrm{D}$“ sind):

- Der Übergang kann dann genommen werden, wenn die aktuelle Zelle σ_1 enthält.
- Der Inhalt der aktuellen Zelle wird mit dem Symbol σ_2 überschrieben.
- Anschließend bewegt sich der Schreib-Lese-Kopf so lange in Richtung D, wie er nicht auf eine Zelle mit Symbol σ_3 trifft. (Er hält also **vor** der Zelle mit Symbol σ_3.) Enthält das Band in Richtung D kein σ_3, gelangt die TM in eine Endlosschleife.

Beispiel:



Nach Ausführung auf der Eingabe *aaab* enthält das Band das Wort *bbbc* und der Schreib-Lese-Kopf befindet sich auf dem ersten *b* von links.

(a) Rechnende Turingmaschine (4 Punkte)

Für $\alpha \in \mathbb{N}_+$ sei

$$f(\alpha) := \begin{cases} \alpha/2, & \text{falls } \alpha \bmod 2 = 0, \\ \texttt{undef}, & \text{sonst.} \end{cases}$$

Erstellen Sie, unter Zuhilfenahme der erweiterten Beschriftungsmöglichkeit, eine Turingmaschine mit **maximal 3 Zuständen**, die f berechnet. Die Eingabe liegt dabei binär kodiert vor, die Ausgabe soll ebenfalls in Binärkodierung erfolgen.

(b) Akzeptierende Turingmaschine (6 Punkte)

Sei $L := \{a^i b^i \mid i \geq 0\} \cup \{a^i b^{2i} \mid i \geq 0\}$.

Erstellen Sie, wieder unter Zuhilfenahme der erweiterten Beschriftungsmöglichkeit, eine (nichtdeterministische) TM mit **maximal 7 Zuständen**, die genau L akzeptiert.

Aufgabe 5: Turing-Vollständigkeit (12 Punkte)

Betrachten Sie die neue Beschriftungsmöglichkeit für Turingmaschinen aus **Aufgabe 4**. Eine um diese Art von Übergängen erweiterte rechnende TM nennen wir *springende Turingmaschine*.

(a) Beweis (6 Punkte)

Zeigen Sie, dass springende TMn nicht mächtiger sind als rechnende TMn.

(b) Busy Bee (6 Punkte)

Eine springende TM mit n Zuständen und $|\Gamma| = 2$, die bei einem anfangs leeren Band die maximale Anzahl an Zustandsübergängen ausführt und hält, heißt *n-Busy Bee*.

Sei $B(n)$ die Anzahl der Zustandsübergänge einer n-Busy Bee. Zeigen Sie, dass $B(n)$ nicht berechenbar ist. Sie dürfen (ohne Beweis) annehmen, dass das Halteproblem für springende TMn unentscheidbar ist.

Achtung: Eine rechnende TM mit n Zuständen lässt sich als springende TM mit i. A. weniger Zuständen simulieren.

Aufgabe 6: Berechenbarkeit und Entscheidbarkeit (12 Punkte)

(a) Berechenbarkeit (6 Punkte)

Sei $g\colon \mathbb{N} \to \{0,1\}$ eine berechenbare Funktion, gegeben durch

$$g(p) \coloneqq \begin{cases} 1, & \text{falls } p \text{ eine Primzahl ist;} \\ 0, & \text{sonst.} \end{cases}$$

Eine TM, die bei Eingabe einer natürlichen Zahl die Funktion g berechnet, heißt *g-TM*. Begründen Sie für die folgende Funktion f (mit Eingabe $x \in \mathbb{N}$), ob sie berechenbar ist:

$$f(x) \coloneqq \begin{cases} 0, & \text{falls } x \text{ die Kodierung } \mathbb{W}(\mathcal{A}) \text{ einer } g\text{-TM } \mathcal{A} \text{ ist, die bei Eingabe } x \text{ hält;} \\ \texttt{undef}, & \text{sonst.} \end{cases}$$

(b) Co-Semi-Entscheidbarkeit (6 Punkte)

Eine deterministische akzeptierende TM $\mathcal{M}$ ist *kürzbar*, wenn sie einen Zustand Z enthält, sodass für jede Eingabe x gilt: Der Berechnungsweg von $\mathcal{M}(x)$ besucht Z nicht.

Zeigen Sie, dass das folgende Problem co-semi-entscheidbar ist:

KürzbareTM	
Gegeben:	Deterministische akzeptierende TM $\mathcal{B}$ mit Eingabealphabet Σ.
Gefragt:	Ist $\mathcal{B}$ kürzbar?

Aufgabe 7: *NP*-Vollständigkeit (16 Punkte)

Eine Juwelierin erhält eine Menge von Perlen, die allesamt zu einer Perlenkette aufgefädelt werden sollen, sodass diese möglichst attraktiv angeordnet sind. Eine Perlenkette $K \coloneqq (k_0, ..., k_{n-1})$ ist eine zyklische Anordnung von mindestens 3 Perlen. Für $0 \leq i < n$ bezeichnen wir $k_{i+1 \bmod n}$ als die *Nachfolgerperle von* k_i.

Die Juwelierin ist im Besitz einer *Attraktivitätsmatrix*, welche für zwei Perlen $p \neq q$ einen Attraktivitätswert $\alpha(p,q) \in \mathbb{N}$ angibt, der erreicht wird, wenn q die Nachfolgeperle von p ist. Die *Attraktivität* einer Kette $K = (k_0, ..., k_{n-1})$ berechnet sich als $\alpha(K) \coloneqq \sum_{i=0}^{n-1} \alpha(k_i, k_{i+1 \bmod n})$.

Perlenkette	
Gegeben:	Menge P von $n \geq 3$ Perlen, Attraktivitätsmatrix $\alpha\colon P \times P \to \mathbb{N}$, natürliche Zahl $A \in \mathbb{N}$.
Gefragt:	Lassen sich alle Perlen aus P so als Kette K anordnen, dass $\alpha(K) \geq A$?

(a) Eingabegröße (4 Punkte)

Geben Sie die Eingabegröße einer Perlenkette-Instanz in Θ-Notation (gekürzt!) an.

(b) Reduktion (12 Punkte)

Zeigen Sie, dass Perlenkette *NP*-schwer ist.

Das in Ihrer Reduktion verwendete Ausgangsproblem $\mathcal{X}$ muss aus der Vorlesung stammen. Geben Sie die Definition von $\mathcal{X}$ nach dem bekannten Schema (Gegeben/Gefragt) an.

Aufgabe 8: Komplexität (10 Punkte)

(a) *Co-NP* (6 Punkte)

Sei $k \in \mathbb{N}$ fest gewählt. Eine aussagenlogische Formel F heißt *k-quasi-wahr*, wenn es maximal k nicht-erfüllende Variablenbelegungen gibt.

Zeigen Sie, dass das folgende Problem k-Quasi-Wahr in *Co-NP* liegt.

k-Quasi-Wahr
Gegeben: Eine aussagenlogische Formel F.
Gefragt: Ist F k-quasi-wahr?

(b) *FPT* (4 Punkte)

Sei $\mathcal{X}$ ein Entscheidungsproblem und n die Kodierungslänge der Eingabe. Wir betrachten verschiedene Kenngrößen k_i, $i \in \{1, 2, 3, 4\}$, der Eingabe (z.B. höchster Zahlwert, maximaler Knotengrad, ...) mit $k_i \geq 1$. Sei $\mathcal{A}$ ein Algorithmus für $\mathcal{X}$ mit Laufzeit $\mathcal{O}(n^{k_1} \cdot {k_2}^{k_3} + k_4)$.

Geben Sie die kleinstmögliche Menge $I \subseteq \{1, 2, 3, 4\}$ an, sodass gilt: Bei konstanten Werten für k_i, $i \in I$, ist das Problem $\mathcal{X}$ in *FPT* in Bezug auf den Parameter $K = \Pi_{i \notin I} k_i$.

Aufgabe 9: Randomisierte Algorithmen (12 Punkte)

(a) Komplexitätsklassen (6 Punkte)

Sei $\mathcal{X}$ ein Entscheidungsproblem. Seien $\mathcal{A}$ und $\mathcal{B}$ zwei randomisierte Polynomialzeitalgorithmen für $\mathcal{X}$, sodass für jede Eingabeinstanz $\mathcal{I}$ gilt:

- Algorithmus $\mathcal{A}$ entscheidet JA-Instanzen immer korrekt und NEIN-Instanzen mit Wahrscheinlichkeit $\geq 1/\text{poly}(|\mathcal{I}|)$ korrekt;
- Algorithmus $\mathcal{B}$ entscheidet NEIN-Instanzen immer korrekt und JA-Instanzen mit Wahrscheinlichkeit $\geq 99\%$ korrekt.

Geben Sie an (ohne Beweis), in welchen der folgenden Komplexitätsklassen das Problem $\mathcal{X}$ nach derzeitigem Stand der Forschung liegt: *BPP*, *Co-RP*, *NL*, *P*, *RP*, *ZPP*.

(b) Analyse (6 Punkte)

Sei $\mathcal{P}$ ein randomisierter Primzahltest: Bei Eingabe einer Primzahl wird immer korrekt `prim` zurückgegeben; bei Eingabe einer zusammengesetzten Zahl wird mit Wahrscheinlichkeit q (wobei $0 < q < 1$) fälschlicherweise `prim` zurückgegeben.

Sei $k \in \mathbb{N}$ konstant gewählt. Der folgende randomisierte Algorithmus $\mathcal{R}_k$ soll bei Eingabe einer Menge $M \subseteq \mathbb{N}$ entscheiden, ob M eine Primzahl enthält.

$\mathcal{R}_k(M)$:

```
wiederhole k-mal:
    wähle ein zufälliges x ∈ M
    if P(x) == prim: return JA
return NEIN
```

Zeigen Sie, welche Fehlerwahrscheinlichkeit Algorithmus $\mathcal{R}_k$ (in Abhängigkeit von q, k und $|M|$) auf NEIN-Instanzen hat.

Viel Erfolg!