Abschlussklausur Theoretische Informatik II

17. Juli 2018

Prof. Dr. Roland Meyer
Elisabeth Neumann
TU Braunschweig
Sommersemester 2018

1. **Bitte ausfüllen:**

   Vorname:

   Nachname:

   Matrikelnummer:

   Unterschrift:

2. Achten Sie darauf, dass Ihre Klausur vollständig ist und getackert bleibt (18 Blätter)!

3. Benutzen Sie **nur das an dieses Blatt angeheftete Papier**. Bei Bedarf können wir weitere Leerblätter austeilen. Wenn der Platz auf der Vorderseite des jeweiligen Aufgabenblatts nicht ausreicht, **machen Sie kenntlich**, wo Sie die Bearbeitung der Aufgabe fortsetzen.

4. Als Hilfsmittel ist **ausschließlich** ein zweiseitig **handgeschriebenes DIN A4-Blatt** erlaubt. Elektronische Geräte müssen während der Klausur ausgeschaltet bleiben. Täuschungsversuche werden mit 0 Punkten gewertet und dem Prüfungsamt gemeldet.

5. Schreiben Sie leserlich und bearbeiten Sie Ihre Klausur **nicht mit Bleistift** und auch **nicht in roter oder grüner Farbe!**

6. Wir werden das Deckblatt während der Klausur auf korrekte Daten überprüfen. Legen Sie dazu Ihren **Studierendenausweis** und einen **amtlichen Lichtbildausweis** bereit.

7. Wir werden den Termin für die Klausureinsicht auf unserer Website bekanntgeben: `tcs.cs.tu-bs.de/teaching/ThInf_SS_2018.html` .

8. Die **Bearbeitungszeit** beträgt **180 Minuten**.

9. Mit **40 Punkten** ist die Klausur **sicher bestanden.**

Punkteverteilung: (wird von den Korrektoren ausgefüllt)

Aufgabe	1	2	3	4	5	6	7	8	9	10	Σ
Max.	10	10	10	10	10	10	10	10	10	10	**100**
Punkte											

1. Konstruktion einer DTM

10 Punkte

Es sei $\Sigma = \{a, b\}$ und $w \in \Sigma^*$ ein Wort. Wir bezeichnen mit $|w|_a$ die Anzahl der a's in w. Analog verwenden wir $|w|_b$ für die Anzahl der b's in w.

Konstruieren Sie eine deterministische Turing-Maschine M, welche die Sprache

$$\mathcal{L} = \{w \in \Sigma^* \mid |w|_a \leqslant |w|_b \text{ oder } 2 \cdot |w|_b = |w|_a\}$$

entscheidet. Geben Sie dabei eine formale Beschreibung von M als Tupel sowie eine ausführliche Erklärung der Arbeitsweise von M an.

Hinweis: Ihre Turing-Maschine darf mehrere Bänder verwenden.

2. TM Analyse

6 + 4 = 10 Punkte

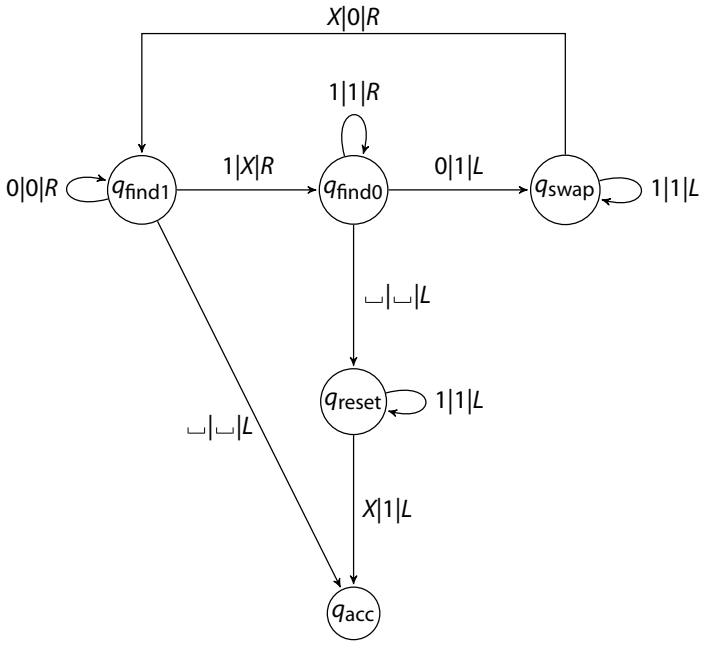
Wir möchten in dieser Aufgabe totale Berechner $M = (Q, \Sigma_1, \Gamma, q_0, \delta, \{q_{\text{acc}}, q_{\text{rej}}\})$ analysieren. Ein totaler Berechner ist analog zu den Berechnern aus der Vorlesung definiert, allerdings muss jede Berechnung auf jeder Eingabe in einem der Haltezustände q_{acc} oder q_{rej} enden. Ähnlich wie in der Vorlesung gilt, dass M die partielle Funktion $f : \Sigma^1 \to_p \Sigma^2$ berechnet, falls für jedes $w \in \Sigma^1$ gilt

$f(w) = w' \in \Sigma^2$ gdw. M mit Eingabe w in q_{acc} hält und w' auf das Band geschrieben hat.

Im Gegensatz zur Vorlesung fordern wir nur, dass der akzeptierende Zustand erreicht wurde und nicht, dass der Kopf von M auf dem ersten Symbol von w' stehen muss. Falls $f(w)$ undefiniert ist, hält M mit Eingabe w in q_{rej}.

Betrachten Sie folgenden totalen deterministischen Berechner $M = (Q, \{0, 1\}, \Gamma, q_{\text{find1}}, \delta, \{q_{\text{acc}}, q_{\text{rej}}\})$, wobei

- $\Gamma = \{\sqcup, 0, 1, X\}$ und
- $Q = \{q_{\text{find1}}, q_{\text{find0}}, q_{\text{swap}}, q_{\text{reset}}, q_{\text{acc}}, q_{\text{rej}}\}$ und
- δ durch folgenden Graphen gegeben ist.



Falls für ein Zustand $q \in Q \backslash \{q_{\text{acc}}, q_{\text{rej}}\}$ und ein Bandsymbol $\gamma \in \Gamma$ keine Transition angegeben ist, ist die Transition gegeben durch $\delta(q, \gamma) = (q_{\text{rej}}, \gamma, N)$. Der Zustand q_{rej} und die fehlenden Transitionen wurden im Graphen zur besseren Lesbarkeit weggelassen.

a) Geben Sie die von M berechnete Funktion $f : \{0,1\}^* \to_p \{0,1\}^*$ an und erklären Sie kurz und informell die Arbeitsweise von M.

b) Analysieren Sie den Zeitverbrauch von dem Berechner, bestimmen Sie also eine Funktion $t\colon \mathbb{N} \to \mathbb{N}$, die den Zeitverbrauch von M beschränkt. (Der Zeitverbrauch für totale Berechner ist analog zu dem Zeitverbrauch für Entscheider definiert.) Begründen Sie Ihre Antwort.

3. Berechenbarkeit

8 + 2 = 10 Punkte

Es sei $\Sigma = \{0, 1\}$. Betrachten Sie die Funktion $loop : (\Sigma^*)^3 \to \Sigma^*$, die wie folgt definiert ist:

$$loop(w_1, w_2, x) = \begin{cases} 1, & \text{falls } M_{w_1}(x) \text{ und } M_{w_2}(x) \text{ beide halten,} \\ w_1, & \text{falls } M_{w_1}(x) \text{ hält und } M_{w_2}(x) \text{ loopt,} \\ w_2, & \text{falls } M_{w_2}(x) \text{ hält und } M_{w_1}(x) \text{ loopt,} \\ 0, & \text{falls } M_{w_2}(x) \text{ und } M_{w_1}(x) \text{ beide loopen.} \end{cases}$$

Dabei sind $w_1, w_2 \in \Sigma^*$ Kodierungen von deterministischen LBAs und $x \in \Sigma^*$ eine Eingabe. Mit $M_{w_i}(x)$ stellen wir die Berechnung von M_{w_i} auf x dar. Sie können davon ausgehen, dass der LBA nur die Zellen verwendet, die anfangs mit dem Eingabewort beschriftet sind (plus Endmarker links und rechts).

Wir sagen, dass die Berechnung von M_{w_i} auf x loopt, falls die Berechnung in eine Schleife gerät, in der kein Haltezustand vorkommt. Da es sich um eine deterministische Turingmaschine handelt, kann diese Schleife nicht mehr verlassen werden und die Berechnung hält nicht.

Hinweis: Falls eine Turingmaschine nur $|x| + 2$ viele Zellen auf dem Band verwendet, ist die Anzahl der Konfigurationen, die während der Berechnung vorkommen können, beschränkt durch

$$|Q| \cdot |\Gamma|^{(|x|+2)} \cdot (|x| + 2).$$

a) Beweisen Sie, dass *loop* berechenbar ist.
   Geben Sie hierzu einen Algorithmus (als Pseudo-Code) an.

b) Was ist die Zeitkomplexität Ihres Algorithmus?

4. TMs mit breitem Kopf

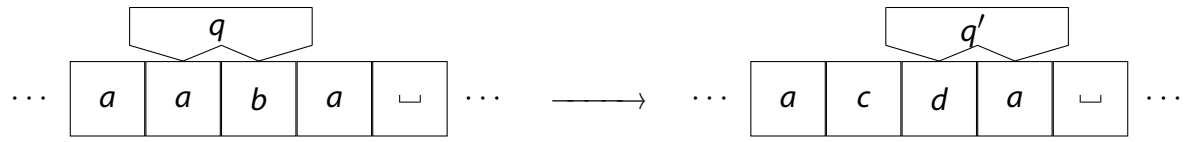
10 Punkte

Eine deterministische Turingmaschine mit breitem Kopf ist eine DTM, die mit ihrem Schreib/-Lesekopf gleichzeitig zwei benachbarte Zellen lesen und beschreiben kann. DTMs mit breiten Kopf sind analog zu den regulären DTMs definiert, haben aber eine Transitionfunktion der folgenden Form.

$$\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, N\}$$

Beispiel:
Eine Transition der Form $q \xrightarrow{ab \mid cd \mid R} q'$ bedeutet, dass

- die TM in Zustand q die Symbole a und b in benachbarten Zellen liest,
- das a durch c sowie das b durch d ersetzt und
- den Kopf nach rechts bewegt und in Zustand q' wechselt.



Bei der Startkonfiguration zeigt der Kopf auf die ersten beiden Symbole der Eingabe.

Zeigen Sie, dass DTMs mit breitem Kopf von herkömmlichen DTMs simuliert werden können. Sei M eine gegebene DTM mit breitem Kopf. Erklären Sie, wie eine DTM M' mit $\mathcal{L}(M') = \mathcal{L}(M)$ konstruiert werden kann.

5. NL-Vollständigkeit

5 + 5 = 10 Punkte

Betrachten Sie das folgende Problem:

Avoid-Reachability (AR)
Gegeben: Ein gerichteter Graph $G = (V, E)$ und Knoten $s, t, v \in V$.
Entscheide: Gibt es in G einen Pfad von s nach t, der v nicht enthält?

Zeigen Sie, dass AR NL-vollständig (bzgl. logspace-many-one-Reduktionen) ist:

a) „Membership“: AR $\in$ NL.

b) „Hardness“: AR ist NL-schwer (bzgl. logspace-many-one-Reduktionen).

6. Entscheidbarkeit

10 Punkte

Betrachten Sie die Sprache aller Kodierungen von Turing-Maschinen, deren Sprachen mindestens ein Wort gerader Länge enthalten:

$$\mathcal{L} = \left\{ w \in \left\{0,1\right\}^* \;\middle|\; \exists x \in \{0,1\}^* : |x| \text{ gerade und } x \in \mathcal{L}\left(M_w\right) \right\}.$$

Beweisen Sie, dass $\mathcal{L}$ nicht entscheidbar ist. Verwenden Sie **nicht** den Satz von Rice.

7. NP-Vollständigkeit

4 + 6 = 10 Punkte

Betrachten Sie das folgende Problem:

Path & Cycle Coverage (PCC)	
Gegeben:	Eine gerichteter Graph $G = (V, E)$.
Entscheide:	Können die Knoten in G durch einen Kreis und einen Pfad überdeckt werden?

Wir möchten also wissen, ob wir die Menge der Knoten V in zwei (möglicherweise leere) Teilmengen $V_C = \{v_1, \ldots, v_k\}$ und $V_P = \{u_1, \ldots, u_n\}$ aufteilen können, so dass

- $v_1 \to v_2 \to \cdots \to v_k \to v_1$ ein Kreis ist, in dem sich kein Knoten außer v_1 wiederholt,
- $u_1 \to u_2 \to \cdots \to u_n$ ein Pfad ist, in dem sich kein Knoten wiederholt, und
- $V = V_C \cup V_P$ mit $V_C \cap V_P = \varnothing$.

Zeigen Sie, dass PCC NP-vollständig (bzgl. Polynomialzeit-Reduktionen) ist:

a) „Membership“: PCC $\in$ NP.

b) „Hardness“: PCC ist NP-schwer (bzgl. Polynomialzeit-Reduktionen).

8. Quiz

(2 + 2 + 2 + 2 + 2) = 10 Punkte

Beantworten Sie die folgenden Fragen. Begründen Sie Ihre Antwort mit einem kurzen Beweis oder einem Gegenbeispiel.

(1) Ist die Funktion $f : \{0, 1\}^* \mapsto \{0, 1\}$ mit $f(w) = \begin{cases} 1, \text{ falls NP = co - NP} \\ 0, \text{ falls NP} \neq \text{co - NP} \end{cases}$ berechenbar?

(2) Wenn $L_1, L_2 \subseteq \{0, 1\}^*$ nicht entscheidbar sind, ist dann auch $L_1 \setminus L_2$ nie entscheidbar?

(3) Gibt es ein Problem aus PSPACE, das EXPSPACE-vollständig ist (bzgl. Polynomialzeit Reduktionen)?
Hinweis: PSPACE $\subsetneq$ EXPSPACE

(4) Ist folgende Argumentation ein Beweis für P $\neq$ NP? Falls nein, wo ist der Fehler in der Argumentation?

a) Angenommen P = NP.

b) Dann gibt es ein $k \in \mathbb{N}$, so dass SAT $\in$ DTIME$\left(\mathcal{O}\left(n^k\right)\right)$.

c) Da jedes Problem aus NP auf SAT reduzierbar ist (bzgl. Polynomialzeit Reduktionen), gilt dann NP $\subseteq$ DTIME$\left(\mathcal{O}\left(n^k\right)\right)$.

d) Laut Annahme P = NP, daher gilt auch P $\subseteq$ DTIME$\left(\mathcal{O}\left(n^k\right)\right)$.

e) Da aber DTIME$\left(\mathcal{O}\left(n^k\right)\right) \subsetneq$ DTIME$\left(\mathcal{O}\left(n^{k+1}\right)\right)$ gilt, ist dies ein Widerspruch zu P $\subseteq$ DTIME$\left(\mathcal{O}\left(n^k\right)\right)$.

(5) Kann man aus dem Satz von Savitch folgern, dass L = NL gilt? Falls nein, argumentieren Sie, warum nicht.

9. Abschluss unter Shuffle

10 Punkte

Sei Σ ein endliches Alphabet und $w, v \in \Sigma^*$ zwei Worte über Σ. Wir definieren die Operation shuffle(w, v) auf w und v wie folgt.

$$\begin{aligned}\text{shuffle}(w, v) = \{x \in \Sigma^* \mid \exists k\, \exists v_1, \ldots, v_k, w_1, \ldots, w_k \in \Sigma^* \\ v = v_1 \ldots v_k \text{ und} \\ w = w_1 \ldots w_k \text{ und} \\ x = w_1.v_1.w_2.v_2 \ldots w_k.v_k\}\end{aligned}$$

Beispielsweise ist shuffle$(ab, cd) = \{abcd, acbd, cdab, cadb, acdb, cabd\}$. Für zwei Sprachen $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$ definieren wir

$$\text{shuffle}(\mathcal{L}_1, \mathcal{L}_2) = \bigcup_{w \in \mathcal{L}_1, v \in \mathcal{L}_2} \text{shuffle}(w, v)$$

Zeigen Sie, dass die Komplexitätsklasse NP abgeschlossen ist unter Shuffle, also dass shuffle$(\mathcal{L}_1, \mathcal{L}_2) \in$ NP, falls $\mathcal{L}_1 \in$ NP und $\mathcal{L}_2 \in$ NP. Erklären Sie, wie ein nicht-deterministischer Entscheider mit polynomieller Laufzeit, der shuffle$(\mathcal{L}_1, \mathcal{L}_2)$ entscheidet, konstruiert werden kann.

10. Nützliche Zustände

4 + 6 = 10 Punkte

Betrachten Sie das folgende Problem:

USEFUL	
Gegeben:	Eine Turingmaschine M und ein Zustand q von M.
Entscheide:	Ist Zustand q nützlich für M?

Ein Zustand q einer Turingmaschine M ist nützlich, falls es eine Eingabe $w \in \{0,1\}^*$ gibt, so dass eine Berechnung von M auf w Zustand q erreicht.

a) Zeigen Sie, dass das Problem USEFUL semi-entscheidbar ist.

b) Zeigen Sie, dass das Problem USEFUL nicht entscheidbar ist.

Hinweis:
Sie können dazu verwenden, dass das Problem

EMPTY	
Gegeben:	Eine Turingmaschine M.
Entscheide:	Ist $\mathcal{L}(M) = \varnothing$?

unentscheidbar ist.