Universität Osnabrück / FB6 / Theoretische Informatik — Prof. Dr. M. Chimani

Informatik D: Einführung in die Theoretische Informatik

Klausur — SoSe 2016 — 11. Juli 2016

Haupttermin, Prüfungsnr. 1007049 — Gruppe: Terence Hill / Mario Girotti

Unbedingt ausfüllen

Matrikelnummer

Studiengang/Abschluss

Fachsemester

Nachname

Vorname

Unterschrift

Identifikator (Beliebiges Wort zur Identifikation im anonymen Notenaushang)

Grundregeln

- Die Bearbeitungszeit der Klausur beträgt **120 Minuten**.
- Sie schreiben diese Klausur **vorbehaltlich** der Erfüllung der **Zulassungsvoraussetzung**. Das heißt: Wir werden Ihre Zulassung vor Korrektur prüfen; die Tatsache, dass Sie die Klausur mitschreiben, bedeutet keine implizite Zulassung.
- Es sind **keine Unterlagen** und auch **keine** anderen **Hilfsmittel** erlaubt.
- Benutzen sie nur dokumentenechten (blauen oder zur Not schwarzen) **Kugelschreiber**! Bleistiftlösungen werden nicht gewertet!
- Es zählt die Antwort, die sich im dafür vorgesehenen Kästchen befindet! Soll eine andere Antwort gewertet werden, so ist diese **eindeutig** zu kennzeichnen!
- Jegliches Schummeln, und auch der Versuch desselben, führt zum Ausschluss von der Klausur und einer Bewertung mit **5,0**.

Wird vom Korrektor/Prüfer ausgefüllt

Aufgabe	1	2	3	4	5	6	7	8	9	10	11	Σ
Punkte (max)	12	10	10	12	12	12	12	16	8	12	20	**136**
Punkte (erreicht)												

Punkte	0..67	68..75	76..83	84..88	89..94	95..99	100..104	105..110	111..115	116..123	124..136
Note	**5,0**	**4,0**	**3,7**	**3,3**	**3,0**	**2,7**	**2,3**	**2,0**	**1,7**	**1,3**	**1,0**

Note:

Aufgabe 1: Chomsky-Hierarchie (12 Punkte)

Zu jeder Sprachfamilie gibt es entsprechende Automatentypen und Grammatiken. Sei V die Menge der Variablen und Σ die Menge der Symbole. Vervollständigen Sie die folgende Tabelle:

Name der Sprachfamilie	Form der Grammatikregeln	Automatentyp(en)
kontextsensitiv		
		NDEA, DEA
	$(V \cup \Sigma)^* V (V \cup \Sigma)^* \to (V \cup \Sigma)^*$	

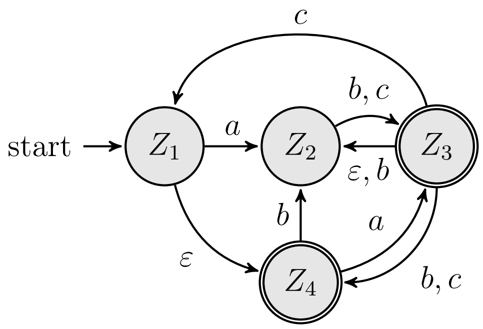
Aufgabe 2: Sprachen klassifizieren (10 Punkte)

Zu welcher Sprachklasse gehören die folgenden Sprachen? Kreuzen Sie dabei *alle* korrekten Antworten an. (Hinweis: Typ 3 = regulär.)

	Typ 3	Typ 2	Typ 1	Typ 0
$\{a^n b^n \mid n \in \mathbb{N} \text{ mit } 2^n = 2n\}$	☐	☐	☐	☐
$\{w \mid w \text{ ist ein regulärer Ausdruck}\}$	☐	☐	☐	☐
$\Sigma^+ \setminus L$, wobei L endlich ist	☐	☐	☐	☐
$\{\mathbb{W}(M) \mid M$ ist eine TM, die auf leerer Eingabe `bier` ausgibt$\}$	☐	☐	☐	☐

Aufgabe 3: Umwandlung: NDEA → DEA (10 Punkte)

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



Aufgabe 4: Deterministisch kontextfreie Sprache (12 Punkte)

Zeigen Sie, dass die durch die folgenden Regeln eingeschränkte Sprache $L \subseteq \Sigma^*$ mit $\Sigma = \{b, d, p, q\}$ *deterministisch* kontextfrei ist:

- Es kommen mindestens so viele q's wie p's vor.
- Jedes Wort endet mit einem b.

Aufgabe 5: Kontextfreie Grammatik **(12 Punkte)**

Geben Sie eine Chomsky-Normalform zu folgender Grammatik an:

$$S \to AS \mid aBcC \qquad B \to CCd \mid A \mid b$$
$$A \to B \mid bb \qquad C \to A \mid B \mid a \mid c$$

Anmerkung zur Korrektur: Sie müssen nicht genau den Schritten aus der Vorlesung folgen (teilweise wäre dies aber empfehlenswert).

Aufgabe 6: Pumping Lemma **(12 Punkte)**

(a) Definition (4 Punkte)

Wie lautet das Pumping Lemma für kontextfreie Sprachen?

Sei L eine kontextfreie Sprache. Dann...

... $z = uvwxy$ mit den Eigenschaften

(1) ,

(2) und

(3) .

(b) Anwendung (8 Punkte)

Vervollständigen Sie den Beweis, dass die Sprache $L := \{a^{\lfloor k/2 \rfloor} b^{\lceil k/3 \rceil} c^j \mid j \geq k \geq 4\}$ nicht kontextfrei ist.

Angenommen, .

Dann wären die Eigenschaften des Pumping Lemmas erfüllt und n die Wortmindestlänge.

Betrachte das Wort $z =$ $\in L$, $|z| \geq n$.

Wegen Eigenschaft kann vwx nur aus höchstens zwei unterschiedlichen Symbolen bestehen, aber nicht gleichzeitig aus .

- Falls das Symbol a in vx enthalten ist, kann man i gemäß Eigenschaft so hoch wählen, dass in uv^iwx^iy .
- Sonst, falls das Symbol b in vx enthalten ist, sind in zu viele a's gegenüber b's.
- Schließlich, falls nur c's in vx enthalten sind, sind .

Aufgabe 7: Rechnende Turingmaschine **(12 Punkte)**

Sei $\Sigma = \{\texttt{a}, \texttt{b}\}$. Geben Sie eine Turingmaschine an, die die Buchstaben der Eingabe aufsteigend sortiert. Die Eingabe ist ein Wort aus Σ^*. Im Definitionsbereich der Funktion liegen nur jene Buchstabenfolgen, bei denen die Buchstaben absteigend sortiert vorliegen. Beachten Sie, dass auch ε eine Eingabe im Definitionsbereich ist.

Beispiel: Aus der Eingabe `bbaaaa` wird die Ausgabe `aaaabb` generiert.

Aufgabe 8: Zusammenhänge **(16 Punkte)**

In jeder Teilaufgabe führt genau ein Kreuz zu einer wahrheitsgemäßen Aussage. Kreuzen Sie die korrekten Aussagen an.
Achtung: Pro Teilaufgabe gibt es $2/0/-1$ Punkte bei einer richtigen/keinen/falschen Antwort! Es gibt jedoch keine negativen Punkte für die gesamte Aufgabe.

Indem wir zeigen, wie wir eine Programmiersprache namens WHITESPACE mit einer Turingmaschine simulieren können, beweisen wir, dass ...

- ☐ ...WHITESPACE mindestens so mächtig ist wie Turingmaschinen.
- ☐ ...Turingmaschinen mindestens so mächtig sind wie WHITESPACE.
- ☐ ...eine Turingmaschine mehr berechnen kann als ein WHITESPACE-Programm.

In ***Co-NP*** liegt ...

- ☐ ...nichts aus ***P***.
- ☐ ...eine echte Teilmenge von ***P***.
- ☐ ...ganz ***P***.

Aus ***P*** = ***NP*** folgt, dass ...

- ☐ ...es keine ***NP***-schweren Probleme gibt.
- ☐ ...alle ***NP***-schweren Probleme deterministisch in polynomieller Zeit lösbar sind.
- ☐ ...auch ***NPC*** = ***NP*** gilt.

Co-NP ist definiert als Komplexitätsklasse, ...

- ☐ ...in der alle Probleme aus ***NP*** liegen, die nicht in ***P*** liegen.
- ☐ ...in der alle Entscheidungsprobleme liegen, deren Komplementprobleme in ***NP*** liegen.
- ☐ ...in der alle Probleme aus ***NP*** liegen, die mit ***NP***-schweren Problemen korrelieren.

Wir betrachten eine SUBSETSUM-Instanz (A, b), wo alle Zahlen $\leq |A|$ sind. Diese Instanz ...

- ☐ ...kann man deterministisch in polynomieller Zeit entscheiden.
- ☐ ...ist stark ***NP***-vollständig.
- ☐ ...ist schwach ***NP***-vollständig.

Um das Theorem von Cook-Levin zu beweisen, ...

- ☐ ...nutzt man eine Reduktion vom Halteproblem auf SAT.
- ☐ ...zeigt man, wie man mit logischen Formeln akzeptierende TMen simuliert.
- ☐ ...reicht es zu zeigen, dass jede logische Formel erfüllbar ist.

Eine Nein-Instanz N für ein Entscheidungsproblem in ***Co-NP*** hat einen Zeugen, der ...

- ☐ ...eine polynomiell große Ja-Instanz ist.
- ☐ ...deterministisch in polynomieller Zeit erzeugt werden kann.
- ☐ ...deterministisch in Polynomialzeit auf Korrektheit überprüft werden kann.

Alle Probleme in ***NP*** sind auch in ...

- ☐ ...***PSPACE***.
- ☐ ...***Co-NP***.
- ☐ ...***NL***.

Aufgabe 9: Turing-Vollständigkeit **(8 Punkte)**

Betrachten Sie die folgende Programmiersprache SchatzWhile:

SchatzWhile-Programme sehen genau wie While-Programme aus. Vor der Ausführung eines SchatzWhile-Programms wird allerdings ein Schatz mit Perlen gefüllt. Wieviele Perlen dies sind, hängt von den Anweisungen im SchatzWhile-Programm ab (siehe unten).

Seien $x_1, x_2, \ldots$ Variablen für natürliche Zahlen. Seien $c \in \mathbb{N}$ Konstanten. Es gibt die folgenden Anweisungen und diese tragen wie folgt zum Schatz bei:

Anweisung	Perlen	Funktionsbeschreibung
$x_i := c$	$c+1$	Zuweisung
$x_i := x_j \pm c$	1	Addition, Subtraktion
while$(x_i \neq c)$ $\{\ \ldots\}$	$i^3 - i^2$	While-Schleife
if $(x_i \neq c)$ $\{\ \ldots\}$ else $\{\ \ldots\}$	i	Bedingtes Ausführen

Die Anweisungen werden mit **;** verkettet. Die Ausführung einer Anweisung kostet immer genau eine Perle, d. h. vor jedem ausgeführten Schritt wird eine aus dem Schatz entnommen. Die Verkettung trägt nichts zum Schatz bei und kostet auch nichts.

Das SchatzWhile-Programm hält an, wenn die aktuelle Anweisung nicht bezahlt werden kann. Es kann – wie ein gewöhnliches While-Programm – schon vorher anhalten.

Beispiel: Das folgende SchatzWhile-Programm hat $1 + 1 + (3^3 - 3^2) + 1 + 1 = 22$ Perlen im Schatz und berechnet $x_2 := 5 \cdot x_1$ für $x_1 \leq 6$ und $x_2 := 35$ sonst:

$$x_2 := 0; \quad x_3 := x_1 + 0; \quad \text{while}(x_3 \neq 0)\ \{\ \ x_2 := x_2 + 5; \quad x_3 := x_3 - 1\ \ \}$$

(a) Definition (4 Punkte)

Definieren Sie Turing-Vollständigkeit.

(b) Turing-Vollständigkeit (4 Punkte)

Ist SchatzWhile Turing-vollständig? Begründen Sie!

Aufgabe 10: Entscheidbarkeit / Algorithmische Fragestellungen (12 Punkte)

DEAs und DLBAs können – analog zu Turingmaschinen, wie aus der Vorlesung bekannt – als Wort dargestellt werden. Für DLBAs $\mathcal{M}$ nutzen wir diese Kodierung $\mathbb{W}(\mathcal{M})$.

Sei A ein DEA mit Zuständen $Z_1, \dots, Z_\alpha$, wobei Z_1 der Startzustand ist und die letzten β Zustände Endzustände sind. Sei $\Sigma := \{a_1, \dots, a_\gamma\}$. Es gibt δ Übergänge $U_1, \dots, U_\delta$. Einen Übergang U_k von Z_i nach Z_j mit dem Symbol a_ℓ kodieren wir als $\mathbb{U}(U_k) := \mathbb{B}(i) \diamond \mathbb{B}(j) \diamond \mathbb{B}(\ell)$. Dann definieren wir $\mathbb{W}_{\text{DEA}}(A) := \mathbb{B}(\alpha) \diamond \mathbb{B}(\beta) \diamond \mathbb{B}(\gamma) \diamond \mathbb{B}(\delta) \diamond \mathbb{U}(U_1) \diamond \cdots \diamond \mathbb{U}(U_\delta)$.

(a) (Un-)Entscheidbarkeit (4 Punkte)

Sei $L := \{\mathbb{W}_{\text{DEA}}(A) \mid A \text{ ist ein DEA, der einen nicht erreichbaren Zustand hat}\}$. Ist L entscheidbar? Wenn ja, wie? Wenn nein, warum nicht?

(b) Halteproblem (4 Punkte)

Wir betrachten beliebige DLBAs $\mathcal{M}$ mit n Zuständen und Bandalphabet Γ. Es existiert eine berechenbare Funktion $f(\mathcal{M}, m) = \mathcal{O}(n \cdot m \cdot |\Gamma|^m)$, die die Anzahl der Schritte angibt, die ein DLBA, der auf einer Eingabe der Länge m hält, maximal machen kann. Warum?

(c) Co-Semi-Entscheidbarkeit (4 Punkte)

Sei $L' := \{\mathbb{W}(\mathcal{M}) \mid \mathcal{M}$ ist ein DLBA, der einen nicht erreichbaren Zustand hat$\}$.
Zeigen Sie, dass L' co-semi-entscheidbar ist. Nutzen Sie dazu die Funktion $f(\mathcal{M}, m)$ aus Aufgabe **(b)**.

Aufgabe 11: *NP*-Vollständigkeit (20 Punkte)

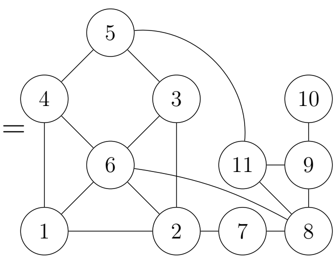
Betrachten Sie das folgende Problem:

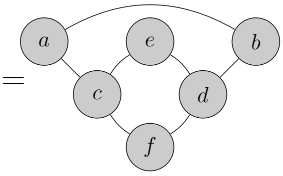
Problem: BinIchDrin (BID)

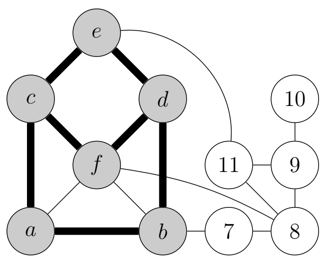
Gegeben: Zwei ungerichtete Graphen $G = (V_G, E_G)$ und $H = (V_H, E_H)$

Frage: Ist H (unter Umbenennung der Knoten) in G enthalten?

Ein Beispiel für eine Ja-Instanz ist die folgende Eingabe:

$G =$ 

$H =$ 

Ja-Instanz, denn: 

Wir wollen im Folgenden zeigen, dass BinIchDrin *NP*-vollständig ist.

Solch ein Beweis besteht typischerweise aus zwei Teilen. Was zeigen wir in Beweisteil 1?

Da es eine offene Frage ist, ob das verwandte Problem GraphIsomorphie in *P* liegt (es ist ein Kandidat für *NPI*), reicht es nicht aus, als Zeuge für BinIchDrin nur die Knoten und Kanten aus H in G ohne Beschriftung zu markieren.

Beschreiben Sie einen geeigneten Zeugen für unser Problem und führen Sie Beweisteil 1 durch.

Beweisteil 1.

Für Beweisteil 2 benötigen wir ein weiteres Problem. Welches wählen Sie?

☐ 3-SAT ☐ GRAPHZUSAMMENHANG ☐ HAMILTONKREIS ☐ SUBSETSUM

Definieren Sie dieses Problem.

Führen Sie nun Beweisteil 2 aus und begründen Sie alle notwendigen Eigenschaften.

Beweisteil 2.

Viel Erfolg!