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

Informatik D: Einführung in die Theoretische Informatik

Klausur — SoSe 2016 — 17. Oktober 2016

Nebentermin, Prüfungsnr. 1007049 — Gruppe: Hase / Igel

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	12	8	6	12	12	16	8	14	26	**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 Formen von Grammatikregeln $x \to y$. Vervollständigen Sie die folgende Tabelle. Geben Sie unter „Form der Grammatikregeln“ genau an, woraus x und y bestehen kann. Nutzen Sie dabei V als die Menge der Variablen und Σ als die Menge der Symbole.

Name der Sprachfamilie	Form der Grammatikregeln	Automatentyp(en)
kontextfrei		
		NDTM, DTM

Welcher Automatentyp entspricht den deterministisch kontextfreien Sprachen?

Wie ist die Laufzeitkomplexität des Wortproblems auf regulären Sprachen?

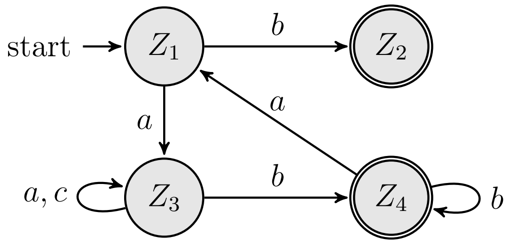
Aufgabe 2: Sprachen klassifizieren **(10 Punkte)**

Zu welcher Sprachfamilie gehören die folgenden Sprachen? Kreuzen Sie dabei die *kleinste* korrekte Antwort an, also die Sprachfamilie, die gerade mächtig genug ist, die entsprechende Sprache zu erkennen. (Hinweis: Typ 3 = regulär.)

	Typ 3	Typ 2	Typ 1	Typ 0
$\Sigma^* \setminus \{wdw^R \mid w \in \{a,b,c\}^*\}$	☐	☐	☐	☐
$\{a^i b^j c^k c^k b^j a^i \mid i,j \in \{5,\dots,23\}, k \in \mathbb{N}\}$	☐	☐	☐	☐
$\{a^i b^j c^k c^k b^j a^i \mid i \in \mathbb{N}, j,k \in \{5,\dots,23\}\}$	☐	☐	☐	☐
$\{J \mid J$ ist eine Ja-Instanz für das Problem $\Pi\}$, wobei Π ein Problem in *EXPSPACE* ist	☐	☐	☐	☐
$\{w \in \{0,1\}^* \mid \lvert w\rvert = n$ und $n \leq S(n)\}$, wobei $S(n)$ die Anzahl der Schritte eines n-Busy-Beavers ist	☐	☐	☐	☐

Aufgabe 3: Umwandlung: DEA → Regulären Ausdruck **(12 Punkte)**

Gegeben sei der folgende DEA. Wandeln Sie ihn – gemäß dem Vorgehen aus der Vorlesung! – in einen regulären Ausdruck um.



Aufgabe 4: Kontextfreie Grammatik (8 Punkte)

Geben Sie eine kontextfreie Grammatik an, die genau die folgende Sprache beschreibt.

$$L = \{a^i b^j c^k d^j e^i f \mid i, j, k \in \mathbb{N},\ i \geq 1,\ k \bmod 2 = 0\}$$

Aufgabe 5: Kellerautomat (6 Punkte)

Gegeben ist die folgende kontextfreie Grammatik:

$$S \to ABB \qquad A \to 1 \mid 00 \qquad B \to 1S \mid A0A$$

Geben Sie einen äquivalenten Kellerautomaten an.

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)

Beweisen Sie, dass die Sprache $L = \{a^i b^j c^k d^j e^k \mid i, j, k \in \mathbb{N}\}$ nicht kontextfrei ist.

Angenommen, ...

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

Sei $\mathbb{U}(\beta)$ die unäre Kodierung einer Zahl $\beta \in \mathbb{N}$.

Geben Sie eine Turingmaschine an, die für ein beliebiges $n \in \mathbb{N}$ und $\alpha_1, \ldots, \alpha_n \in \mathbb{N}_+$, aus der Eingabe $\mathbb{U}(\alpha_1)\square\mathbb{U}(\alpha_2)\square\cdots\square\mathbb{U}(\alpha_n)$ die Ausgabe $\mathbb{U}(\alpha_1 + \alpha_2 + \ldots + \alpha_n)$ berechnet.

Im Definitionsbereich liegen genau die Eingaben mit $n \geq 2$.

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.

Für zwei beliebige kontextfreie Grammatiken G_1, G_2 ist es unentscheidbar, ...

- ☐ ...ob $\mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ kontextfrei ist.
- ☐ ...ob $\mathcal{L}(G_1) = \varnothing$ und $\mathcal{L}(G_2) = \varnothing$ gilt.
- ☐ ...ob $\mathcal{L}(G_1) \cap \mathcal{L}(G_2)$ kontextfrei ist.

Aus $\mathsf{NP} \subseteq \mathsf{P}$ würde folgen, dass ...

- ☐ ...DTMen mindestens so viel berechnen können wie NDTMen.
- ☐ ...DTMen NDTMen in Polynomialzeit simulieren können.
- ☐ ...NDTMen mindestens so viel berechnen können wie DTMen.

Eine (partielle) Funktion $g\colon \mathrm{Def} \to \mathbb{N}$ heißt berechenbar, falls eine endliche algorithmische Beschreibung existiert, die ...

- ☐ ...für jedes $x \in \mathrm{Def}$ den Wert $g(x)$ berechnet und sonst nicht terminiert.
- ☐ ...mit einer Turingmaschine ausführbar ist und bei `undef` in eine Endlosschleife geht.
- ☐ ...für jedes $X \in \mathbb{N}$ nicht terminiert und sonst co-semi-entscheidbar ist.

Ein Forscher findet ein Turing-vollständiges mathematisches Berechnungsmodell Υ und möchte zeigen, dass es echt mächtiger ist als eine NDTM. Dazu muss er ...

- ☐ ...zeigen, dass Υ das Halteproblem entscheiden kann.
- ☐ ...ein Beispiel finden, dass von Υ, aber nicht von einer NDTM berechnet werden kann.
- ☐ ...zeigen, wie Υ NDTMen simulieren kann.

Eine Sprache, die rekursiv aufzählbar ist, ...

- ☐ ...ist entscheidbar.
- ☐ ...kann co-semi-entscheidbar sein.
- ☐ ...ist unentscheidbar.

Die Kolmogorov-Komplexität $K_P(w)$ gibt an, wie lang ...

- ☐ ...ein in P geschriebenes w-erzeugendes Programm mindestens sein muss.
- ☐ ...das kürzeste w ist, das das Programm P ausgibt.
- ☐ ...die kürzeste Beschreibung von $P(w)$ ist.

Wir nehmen an, dass $\mathsf{P} \neq \mathsf{NP}$. Eine Instanz (mit Eingabegröße n) eines schwach NP-vollständigen Problems ist in polynomieller Zeit lösbar, wenn ...

- ☐ ...alle maximal n vorkommenden Zahlen durch ein binäres Polynom beschränkt sind.
- ☐ ...n durch ein Polynom der größten vorkommenden Zahl beschränkt ist.
- ☐ ...alle vorkommenden Zahlen polynomiell in n beschränkt sind.

Aus $\mathsf{NPI} = \varnothing$ würde folgen, dass ...

- ☐ ...$\mathsf{P} = \mathsf{NL}$.
- ☐ ...$\mathsf{Co\text{-}NP} = \mathsf{NP}$.
- ☐ ...$\mathsf{EXPTIME} = \mathsf{NPI}$.

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

Betrachten Sie die folgende Programmiersprache LoopAHop:

Variablenbezeichner sind beliebige Wörter aus $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}, \texttt{0}, \texttt{1}, \ldots, \texttt{9}\}^+ \setminus \{\texttt{hop}\}$. Die Variablen nehmen Werte von natürlichen Zahlen an.

Seien $c \in \mathbb{N}$ beliebige Konstanten. Es gibt die folgenden LoopAHop-Anweisungen:

var `:` c	weist der Variablen *var* den Wert c zu,
var1 `:` *var2*	weist der Variablen *var1* den Wert der Variablen *var2* zu,
var `+` c	erhöht den Wert der Variablen *var* um c,
var `-` c	verringert den Wert der Variablen *var* um c,
var1 `+` *var2*	erhöht den Wert der Variablen *var1* um den Wert der Variablen *var2*,
var1 `-` *var2*	verringert den Wert der Variablen *var1* um den Wert der Variablen *var2*,
$\mathcal{A}$ `?` *var*	führt Anweisung $\mathcal{A}$ nur aus, wenn der Wert der Variablen *var* genau 0 ist,
`{ ... }`	führt die Anweisung(en) zwischen den Klammern in einer (∞-)Schleife aus,
`hop`	springt aus der innersten durch `{` und `}` definierten Schleife heraus, oder beendet das Programm, falls keine umgebenden Schleifenklammern existieren.

Eine Verkettung von Anweisungen findet durch Leerzeichen oder Zeilenumbruch zwischen den Anweisungen statt.

Beispiel: Sei in `i` ein Wert > 0 gespeichert. Das folgende LoopAHop-Programm setzt `fib` auf die `i`-te Fibonaccizahl:

```
fib:1  i-1  hop?i  b:0  {  a:b  b:fib  fib+a  i-1  hop?i  }
```

Zeigen oder widerlegen Sie, dass LoopAHop Turing-vollständig ist.

Aufgabe 10: Leerheitsproblem **(14 Punkte)**

(a) Reguläre Sprachen (4 Punkte)

Sei L eine reguläre Sprache, d. h. es gibt einen DEA A mit $L = \mathcal{L}(A)$. Ist das Leerheitsproblem für L entscheidbar? Wenn ja, wie? Wenn nein, warum nicht?

(b) Kontextsensitive Sprachen (6 Punkte)

Sei L eine kontextsensitive Sprache, d. h. es gibt einen NLBA A mit $L = \mathcal{L}(A)$. Das Leerheitsproblem ist für L nicht entscheidbar. Zeigen Sie, dass diese Frage co-semi-entscheidbar ist.

Hinweis: Muss A gewisse Eigenschaften erfüllen?

(c) Unentscheidbare Folgen (4 Punkte)

Basierend auf dem Wissen, dass das Leerheitsproblem unentscheidbar ist, folgt direkt, dass für zwei gegebene Sprachen L_1, L_2 die Fragen

(1) $L_1 = L_2$ und (2) $L_1 \cap L_2 = \varnothing$

unentscheidbar sein müssen. Wie folgt dies?

Aufgabe 11: *NP*-Vollständigkeit (26 Punkte)

(a) Reduktion (8 Punkte)

Was versteht man unter einer Reduktion bei einem *NP*-Vollständigkeitsbeweis?
Was zeigt man damit?

(b) Vertex Cover (4 Punkte)

Definieren Sie das Problem Vertex Cover.

(c) Anwendung (10 Punkte)

Ada ist Programmiererin und hat eine Menge $\mathcal{M}$ von Modulen $M_1, \ldots, M_n$, die getestet werden sollen. Dazu existieren eine Menge $\mathcal{T}$ an Testprogrammen $T_1, \ldots, T_m$. Ein Testprogramm T_i testet aber nicht nur genau ein Modul, sondern gegebenenfalls mehrere; also die Modulmenge $\mu_{T_i} \subseteq \mathcal{M}$.

Ada möchte nun, dass ihr Kollege Kim alle Module testet. Um Zeit zu sparen, möchte Kim möglichst wenige Testprogramme ausführen, aber natürlich so, dass alle Module getestet werden.

Definieren Sie das zugehörige Entscheidungsproblem zum geschilderten Optimierungsproblem *möglichst mathematisch-formal*.

Zeigen Sie mithilfe von VERTEX COVER, dass Kims Problem $\mathbf{NP}$-schwer ist.

(d) Optimierungsproblem lösen (4 Punkte)

Sei $\mathcal{A}$ ein Algorithmus, der Kims Entscheidungsproblem in Zeit $T(n, m)$ löst. Wie löst man damit Kims Optimierungsproblem mit multiplikativem logarithmischen Mehraufwand?

Viel Erfolg!