Institut für Informatik der Ludwig-Maximilians-Universität München
Prof. Dr. David Sabel
Stephan Barth
Sebastian Sturm

SoSe 2019
24.09.2019

Lösung zur Klausur zur Vorlesung

Formale Sprachen und Komplexität

Die Bearbeitungszeit beträgt 120 Minuten. Hilfsmittel sind nicht erlaubt, auch der Besitz ausgeschalteter elektronischer Geräte wird als Betrug gewertet. Schreiben Sie Ihren vollständigen Namen und Ihre Matrikelnummer deutlich lesbar auf dieses Deckblatt, sowie Ihren Namen in die Kopfzeile auf **jedem Blatt** der Klausurangabe! Geben Sie alle Blätter ab, lassen Sie diese zusammengeheftet! Verwenden Sie nur dokumentenechte Stifte und weder die Farben rot noch grün.

Kontrollieren Sie, ob Sie alle Aufgabenblätter erhalten haben. Aufgabenstellungen befinden sich auf den Seiten 1–13. Sie dürfen die Rückseiten für Nebenrechnungen nutzen. Falls Sie die Rückseiten für Antworten nutzen, so markieren Sie klar, was zu welcher Aufgabe gehört und schreiben Sie in der entsprechenden Aufgabe, wo alle Teile Ihrer Antwort zu finden sind. Streichen Sie alles durch, was nicht korrigiert werden soll!

Lesen Sie die Aufgabenstellungen vollständig durch, bevor Sie mit der Bearbeitung der Aufgaben beginnen. Es gibt 8 unterschiedlich gewichtete Aufgaben zu insgesamt 120 Punkten. Mit 60 Punkten haben Sie sicherlich bestanden. Die Teilaufgaben können unabhängig voneinander bearbeitet werden.

Mit Ihrer Unterschrift bestätigen Sie, dass Sie zu Beginn der Klausur in ausreichend guter gesundheitlicher Verfassung sind, und diese Klausurprüfung verbindlich annehmen.

Nachname:

Vorname:

Matrikelnummer:

Studiengang:

Die Klausur soll gewertet werden: ☐ **JA** ☐ **NEIN**

Hinweis: Ist keines der beiden Felder angekreuzt, so wird **JA** angenommen.

Hiermit erkläre ich die Richtigkeit der obigen Angaben:

(Unterschrift)

Die folgende Tabelle **NICHT** ausfüllen:

Aufgabe	1	2	3	4	5	6	7	8	$\sum$	Bonus	Note
Punkte	18	12	12	18	15	15	18	12	**120**		
erreicht											

Lösung Aufgabe 1 (Allgemeines): **(18 Punkte)**

Pro Frage gibt es genau eine richtige Antwort.

Punkte in dieser Aufgabe werden nach folgender Tabelle vergeben. Eine Antwort, bei der 0 oder mindestens 2 Kästchen angekreuzt sind, zählt als falsch.

Richtige Antworten	0	1	2	3	4	5	6	7	8	9	10	11	12
Erzielte Punkte	0	0	0	0	2	4	6	8	10	12	14	16	18

Die primitiv rekursiven Funktionen umfassen alle

- [ ] WHILE-berechenbaren Funktionen.
- [X] LOOP-berechenbaren Funktionen.
- [ ] μ-rekursiven Funktionen.
- [ ] Turingberechenbaren Funktionen.

Die Sprache $L = \{a^n b^n \mid n \in \mathbb{N}\}$

- [ ] ist nicht kontextfrei.
- [ ] ist endlich.
- [ ] hat einen endlichen Nerode-Index.
- [X] ist nicht regulär.

Deterministische Kellerautomaten, die mit Endzuständen akzeptieren, erkennen

- [ ] genau die Typ 2-Sprachen.
- [ ] die gleichen Sprachen, wie deterministische Kellerautomaten, die bei leerem Keller akzeptieren.
- [X] insbesondere alle regulären Sprachen.
- [ ] genau die Sprachen, die von NFAs erkannt werden.

Sei $a : (\mathbb{N} \times \mathbb{N}) \to \mathbb{N}$ die Ackermannfunktion, $bin : \mathbb{N} \to \{0,1\}^*$ die Binärkodierung einer Zahl und M die durch $w_M \in \{0,1\}^*$ kodierte Turingmaschine. Eine entscheidbare Sprache ist

- [ ] $\{w_M \in \{0,1\}^* \mid M \text{ hält bei leerer Eingabe}\}$
- [ ] $\{w_M \in \{0,1\}^* \mid M \text{ hält bei Eingabe } bin(a(|w_M|, |w_M|)\ \}$
- [ ] $\{w_M \in \{0,1\}^* \mid M \text{ hält nicht bei leerer Eingabe}\}$
- [X] $\{w_M \in \{0,1\}^* \mid M \text{ hält bei leerer Eingabe nach höchstens } a(|w_M|, |w_M|) \text{ Schritten}\}$

Welcher dieser Formalismen beschreibt eine andere Sprachklasse als die anderen drei Formalismen?

- [ ] Reguläre Ausdrücke
- [X] Typ 1-Grammatiken
- [ ] Reguläre Grammatiken
- [ ] Nichtdeterministische endliche Automaten

Welche der folgenden Eigenschaften trifft für die Grammatik
$G = (\{S, A, B\}, \{a, b\}, \{S \to AB \mid BA, A \to a, B \to b, AA \to BB\}, S)$ zu?

- [X] G ist in Kuroda-Normalform.
- [ ] G enthält Einheitsproduktionen.
- [ ] G ist kontextfrei.
- [ ] G enthält Epsilonproduktionen.

Die Aufgabe wird auf dem nächsten Blatt fortgesetzt.

Fortsetzung von Aufgabe 1:

Welche Aussage stimmt?

- [ ] Jede Typ 0-Sprache ist entscheidbar.
- [ ] Jede Typ 0-Sprache ist unentscheidbar.
- [X] Jede entscheidbare Sprache ist vom Typ 0.
- [ ] Jede unentscheidbare Sprache ist vom Typ 0.

Aus welcher der folgenden Eigenschaften folgt, dass eine Sprache L über dem Alphabet Σ regulär ist?

- [ ] Es gibt eine Zahl $n > 0$, sodass jedes Wort $z \in L$ mit $|z| \geq n$, als $z = uvw$ geschrieben werden kann, so dass gilt: i) $|uv| \leq n$, ii) $|v| \geq 1$, iii) für alle $i \geq 0 : uv^iw \in L$
- [ ] Für jede Zahl $n > 0$, gibt es ein Wort $z \in L$ mit $|z| \geq n$ und für jede Zerlegung $z = uvw$ mit $|uv| \leq n$ und $|v| \geq 1$ gibt es ein $i \geq 0$, sodass $uv^iw \notin L$
- [ ] Es gibt eine Zahl $n > 0$, sodass jedes Wort $z \in L$ mit $|z| \geq n$, als $z = uvwxy$ geschrieben werden kann, so dass gilt: i) $|vwx| \leq n$, ii) $|vx| \geq 1$, iii) für alle $i \geq 0 : uv^iwx^iy \in L$
- [X] Es gibt eine Zahl $n > 0$, sodass für jedes Wort $z \in \Sigma^*$ mit Mindestlänge n (d.h. $|z| \geq n$) gilt: für alle $i > 0 : z \notin L^i$.

Sei L eine kontextfreie Sprache.

- [ ] Dann ist der Nerode-Index von L unendlich.
- [X] Dann gibt es eine Teilsprache $L' \subseteq L$, sodass L' regulär ist.
- [ ] Dann ist $\overline{L}$ keinesfalls eine reguläre Sprache.
- [ ] Dann verletzt L die Pumpingeigenschaft für reguläre Sprachen.

Seien L und M Sprachen und $M \in \mathcal{NP}$. Aus welcher der folgenden Eigenschaften folgt, dass M $\mathcal{NP}$-vollständig ist.

- [X] $L \leq_p M$ und L ist $\mathcal{NP}$-hart.
- [ ] $L \leq_p M$ und $L \in \mathcal{NP}$.
- [ ] $M \leq_p L$ und L ist $\mathcal{NP}$-hart.
- [ ] $M \leq_p L$ und $L \in \mathcal{NP}$.

Welche der folgenden Sprachen ist von einem anderen Typ als die anderen drei Sprachen?

- [ ] $L(a^*b \mid ba^*)$
- [ ] $L(G)$ wobei $G = (\{S, B\}, \{a, b\}, \{S \to aS \mid B, B \to Bb \mid b\}, S)$
- [ ] $\{a^ib^ic^i \mid i \leq 100\}$
- [X] $L(G')$ wobei $G' = (\{S, B\}, \{a, b\}, \{S \to aB, B \to Sb \mid b\}, S)$

Ein minimaler NFA für eine Sprache L ist ein NFA, der L erkennt und eine minimale Anzahl an Zuständen besitzt. Die Anzahl der Zustände eines minimalen NFA ist immer

- [ ] größer als der Nerode-Index.
- [ ] das Doppelte des Nerode-Index.
- [X] kleiner oder gleich dem Nerode-Index.
- [ ] hat mit dem Nerode-Index nichts zu tun.

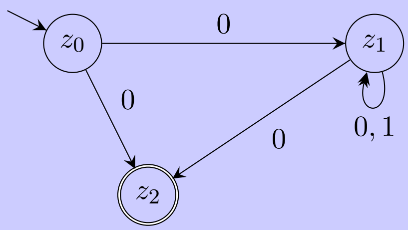
Lösung Aufgabe 2 (Formalismen für Reguläre Sprachen): **(12 Punkte)**

Punkteverteilung: a) 5 Punkte b) 2 Punkte c) 5 Punkte

Sei F die formale Sprache über dem Alphabet $\{0, 1\}$, sodass F genau die Worte enthält, die mit einer 0 enden und nicht mit einer 1 beginnen, wobei das leere Wort nicht in F enthalten ist.

a) Geben Sie einen NFA A als Zustandsgraph an, sodass $L(A) = F$ gilt.

LÖSUNG:



Bewertung: je 1 P:

- 0 akzeptieren
- mit 0 beginnen
- mit 0 enden
- beliebig viele 0,1 dazwischen
- Reihenfolge von 0,1 uneingeschränkt

Abzüge bei fehlenden Start- / Endzustand.

b) Geben Sie einen regulären Ausdruck α an, sodass $L(\alpha) = F$ gilt. **LÖSUNG:** 0 | 0(0 | 1)*0

Bewertung: 1 Punkt für 0 alleine, 1 Punkt für 0(0|1)*0.

c) Geben Sie eine reguläre Grammatik G an, sodass $L(G) = F$ gilt.

LÖSUNG: $G = (V, \Sigma, P, S)$, wobei

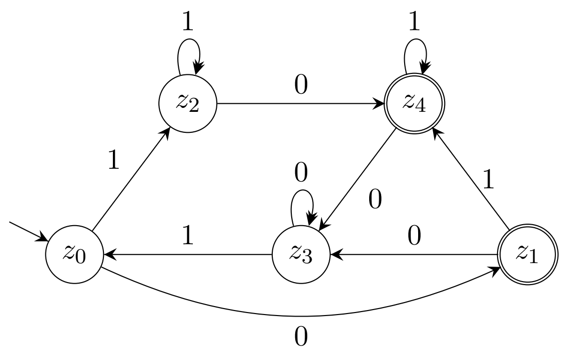
- $V = \{S\}$
- $\Sigma = \{0, 1\}$
- $P = \{S \to 0 \mid 0A,$
  $A \to 0 \mid 0A \mid 1A\}$

Bewertung: 3 Punkte für Formalität, 2 für Produktionsregeln.

Lösung Aufgabe 3 (DFA-Minimierung): (12 Punkte)

Punkteverteilung: a) 10 Punkte b) 2 Punkte

Der DFA A über dem Alphabet $\Sigma = \{0, 1\}$ ist durch den folgenden Zustandsgraphen gegeben:

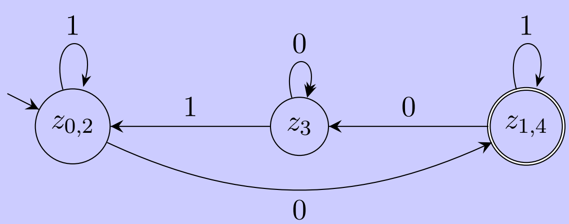


a) Berechnen Sie einen Minimalautomaten B mit $L(A) = L(B)$.
Geben Sie den Rechenweg an (Minimierungstabelle).

LÖSUNG:

z_1	X_E			
z_2		X_E		
z_3	$1[z_1 z_3]$	X_E	$1[z_4 z_3]$	
z_4	X_E		X_E	X_E
	z_0	z_1	z_2	z_3

Daher können z_0 und z_2 sowie z_1 und z_4 jeweils verschmolzen werden. Das ergibt:



6 Punkte für die Tabelle (4 für die Initialisierung, 2 für die Iteration)
4 Punkte für den DFA

b) Wie lautet der Nerode-Index von $L(A)$? Begründen Sie Ihre Antwort.

LÖSUNG: Der Nerode-Index ist 3, da er genau der Zustandsanzahl des Minimalautomaten entspricht und dieser 3 Zustände hat.

1 Punkt für 3 und 1 Punkt für die korrekte Begründung. Folgefehler: Wenn bei a) Automat mehr Zustände hat, dann und sonst alles stimmt, kein Abzug.

Lösung Aufgabe 4 (Akzeptanz von endlichen Automaten): **(18 Punkte)**

Punkteverteilung: a) 7 Punkte b) 7 Punkte c) 4 Punkte

Spezielle deterministische endliche Automaten (SDFAs) sind DFAs, die das Alphabet $\{a\}$ verwenden und die genau einen Endzustand haben:

Definition: SDFA

Ein SDFA ist ein Tupel $A = (Z, \Sigma, \delta, z_0, z_E)$, wobei Z die Zustandsmenge, $\Sigma = \{a\}$ das Alphabet, $\delta : Z \times \Sigma \to Z$ die totale Übergangsfunktion, $z_0 \in Z$ der Startzustand und $z_E \in Z$ der einzige Endzustand ist.
Sei $\widehat{\delta} : (Z \times \Sigma^*) \to Z$ definiert als $\widehat{\delta}(z, aw) := \widehat{\delta}(\delta(z, a), w)$ und $\widehat{\delta}(z, \varepsilon) := z$.
Dann ist die vom SDFA A akzeptierte Sprache definiert als $L(A) := \{w \mid \widehat{\delta}(z_0, w) = z_E\}$.

Beweisen oder widerlegen Sie die folgenden Aussagen:

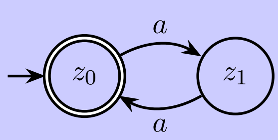
a) Es gibt einen SDFA A mit $|Z| = 2$, sodass $a \in L(A)$ und $aa \in L(A)$, aber $aaa \notin L(A)$.

LÖSUNG: Die Aussage ist falsch. O.B.d.A. sei $Z = \{z_0, z_1\}$. Da A die Worte a und aa akzeptiert werden und es nur genau einen akzeptierenden Zustand gibt, muss gelten $\delta(z_0, a) = z_i$ und $\widehat{\delta}(z_0, aa) = \delta(\delta(z_0, a), a) = \delta(z_i, a) = z_i$ mit $z_E = z_i$. Dann gilt jedoch auch $\widehat{\delta}(z_0, aaa) = \widehat{\delta}(\delta(z_0, a), aa) == \widehat{\delta}(z_i, aa) = \widehat{\delta}(\delta(z_i, a), a) == \widehat{\delta}(z_i, a) = \widehat{\delta}(\delta(z_i, a), \varepsilon) = \delta(z_i, a) = z_i$. Daher akzeptiert A das Wort aaa.

2 Punkte für „falsch". 5 Punkte für den Beweis.

b) Es gibt einen SDFA B mit $|Z| = 2$, sodass $a \notin L(B)$ und $aaa \notin L(B)$, aber $aaaa \in L(B)$.

LÖSUNG: Die Aussage ist wahr. Der SDFA B kann konstruiert werden als



2 Punkte für „richtig". 5 Punkte für den Beweis (den Automat)

c) Seien C und D beliebige SDFAs und $L(C)$ und $L(D)$ die akzeptierten Sprachen. Dann kann man einen SDFA konstruieren, der genau die Sprache $L(C) \cap L(D)$ akzeptiert.

LÖSUNG: Ein solcher SDFA kann als Produktautomat konstruiert werden. Sei $C = (Z_C, \{a\}, \delta_C, z_{C,0}, z_{C,E})$ und $D = (Z_D, \{a\}, \delta_D, z_{D,0}, z_{D,E})$. Konstruiere den Automaten $M = (Z_C \times Z_D, \{a\}, \delta_M, (z_{C,0}, z_{D,0}), (z_{C,E}, z_{D,E}))$ mit $\delta_M((z, z'), a) = (\delta_C(z, a), \delta_D(z', a))$.

Zunächst zeigen wir $\widehat{\delta_M}((z,z'),w) = (\widehat{\delta_C}(z,w),\widehat{\delta_D}(z',w))$ per Induktion über $|w|$: Die Induktionsbasis gilt da $\widehat{\delta_M}((z,z'),\varepsilon) = (z,z') = (\widehat{\delta_C}(z,\varepsilon),\widehat{\delta_D}(z',\varepsilon))$. Für den Induktionsschritt sei $w = au$ und (als Induktionsannahme) gelte die Behauptung für u. Dann gilt $\widehat{\delta_M}((z,z'),au) = \widehat{\delta_M}(\delta_M((z,z'),a),u) = \widehat{\delta_M}((\delta_C(z,a),\delta_D(z',a)),u) = (\widehat{\delta_C}(\delta_C(z,a),u),\widehat{\delta_D}(\delta_D(z',a),u)) = (\widehat{\delta_C}(z,au),\widehat{\delta_D}(z',au))$.

Nun zeigen wir: M akzeptiert $L(C) \cap L(D)$: Sei $w \in L(C) \cap L(D)$, dann gilt $\widehat{\delta_C}(z_{C,0},w) = z_{C,E}$ und $\widehat{\delta_D}(z_{D,0},w) = z_{D,E}$. Mit dem vorher gezeigten folgt $\widehat{\delta_M}((z_{C,0},z_{D,0}),w) = (z_{C,E},z_{D,E})$ und daher $w \in L(M)$.

Sei $w \notin L(C) \cap L(D)$, dann gilt $\widehat{\delta_C}(z_{C,0},w) = z_1$ und $\widehat{\delta_D}(z_{D,0},w) = z_2$, wobei z_1 oder z_2 kein Endzustand ist. Mit dem vorher gezeigten folgt $\widehat{\delta_M}((z_{C,0},z_{D,0}),w) = (z_1,z_2)$ und daher $w \notin L(M)$.

Schließlich kann man noch nachprüfen, dass M die Eigenschaften eines SDFA erfüllt.

2 Punkte für „richtig“. 2 Punkte für den Beweis.

Lösung Aufgabe 5 (CYK-Algorithmus): **(15 Punkte)**

Punkteverteilung: a) 10 Punkte b) 5 Punkte

Sei $G = (\{S, X, Y, A, B\}, \{a, b\}, P, S)$ mit

$$P = \{ \quad S \to BX \mid YA \mid AA \mid BB \mid a \mid b, \\ X \to SB, \\ Y \to AS, \\ A \to a, \\ B \to b \quad \}$$

a) Entscheiden Sie, ob

$$aababaa \in L(G)$$

gilt, indem Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus) ausführen. Geben Sie die dabei entstehende Tabelle und die Ausgabe des Algorithmus an und dokumentieren Sie Ihren Rechenweg.

LÖSUNG: Sei $w = aababaa$. Die folgende Tabelle zeigt $V[i, j]$, wobei diese Menge, genau die Nichterminale enthält, die das Wort $w[i]w[i+1]\cdots w[i+j]$ herleiten können.

	a	a	b	a	b	a	a
j/i	1	2	3	4	5	6	7
1	S, A	S, A	S, B	S, A	S, B	S, A	S, A
2	S, Y	X, Y		X, Y		S, Y	
3	X	S	S	S			
4	Y	Y, X					
5		S					
6	Y						
7	S						

Da $S \in V[1, 7]$ kann S das Wort $aababaa$ herleiten und der Algorithmus liefert True zurück.

- 2 Punkte bei einem korrekten Endergebnis, d.h. wenn in der untersten Zelle ein S steht;
- 2 Punkte bei einer richtigen Schlussfolgerung, unabhängig davon, ob das Ergebnis korrekt ist, d.h. „S in der untersten Zelle $\implies$ $aababaa \in L(G)$“ ist sicherlich korrekt, aber „$aababaa \notin L(G)$“ bringt auch 2 Punkte, wenn S nicht in seiner/ihrer untersten Zelle liegt;
- Die übrigen 6 Punkte werden wie folgt vergeben: die 28 Zellen von Zeilen 1-7 der Musterlösung werden mit den entsprechenden 28 Zellen der Klausurantwort quasi logisch XOR-verknüpft, ein „Abstand“ ergibt sich aus der Anzahl der diskrepanten Zellen, dann: 6 Punkte bei einem Abstand 0, also alle Zwischenschritte sind identisch zur Musterlösung; Je 1 Punktabzug bei einer inkorrekten Zelle, dies betrifft sowohl fehlende als auch überschüssige Nichtterminalsymbole, ebenso die unterste Zelle, wenn sie neben S noch was anderes enthält... Wenn mindestens 6 Zellen inkorrekt sind, wird kein Punkt in diesem Bereich vergeben; Bei mehr als 6 inkorrekten Zellen erfolgt kein extra Punktabzug.

b) Geben Sie an, welche Teilworte von *aababaa* ebenfalls in $L(G)$ liegen, indem Sie die Tabelle aus dem vorherigen Aufgabenteil verwenden (und keine neuen Tabellen berechnen). Erläutern Sie für jedes der Worte, warum sie schließen können, dass es in der Sprache $L(G)$ liegt.

Alternativ dürfen Sie diese Aufgabe bearbeiten (z.B. wenn Sie Aufgabenteil (a) nicht bearbeitet haben), indem Sie *allgemein* beschreiben, wie man den CYK-Algorithmus abändern kann, um herauszufinden, welche Teilworte für ein gegebenes Wort w in der Sprache $L(G)$ liegen.

LÖSUNG: Man muss nur die Tabelleneinträge nehmen, die S enthalten und die Teilworte entsprechend ablesen. Konkret schreiben wir bei allen Einträgen, die S enthalten, das entsprechende Teilwort $w[i], ..., w[i+j]$ dazu:

	a	a	b	a	b	a	a
j/i	1	2	3	4	5	6	7
1	$S, A : a$	$S, A : a$	$S, B : b$	$S, A : a$	$S, B : b$	$S, A : a$	$S, A : a$
2	$S, Y : aa$	X, Y		X, Y		$S, Y : aa$	
3	X	$S : aba$	$S : bab$	$S : aba$			
4	Y	Y, X					
5		$S : ababa$					
6	Y						
7	$S : aababaa$						

Das ergibt die Worte, a, b, aa, aba, bab, $ababa$, $aababaa$

- 2 Punkte, wenn die Lösung die Tabelle aus Teilaufgabe a) verwendet oder deutlich darauf hingewiesen hat, dass man einfach alle Tabelleneinträge, die S enthalten, nehmen und die Teilworte entsprechend ablesen kann, oder ähnliches;
- Die übrigen 3 Punkte werden wie folgt vergeben: 3 Punkte, wenn alle angegebenen herleitbaren Teilworte korrekt sind; Je 1 Punktabzug bei einem inkorrekten Wort, dies betrifft sowohl falsch positiv als auch falsch negativ genannte Worte; Nur 1 Punktabzug, wenn sowohl a als auch b fehlen; Kein Punktabzug, wenn aababaa fehlt; Kein Punktabzug, wenn erkennbar ein Folgefehler von Teilaufgabe a) ist und sonst alles ordentlich gemacht ist (Ermessensentscheidung); Wenn mindestens 3 Worte inkorrekt sind, wird kein Punkt in diesem Bereich vergeben; Bei mehr als 3 inkorrekten Worten erfolgt kein extra Punktabzug.
- Sonderfall: Wenn die Lösung kein einziges Wort angibt und nur sehr allgemein beschreibt, wie man alle herleitbaren Teilworte eines gegebenen Wortes finden kann, können auch unmittelbar 5 Punkte vergeben werden, wenn die Beschreibung das Wesentliche trifft und fehlerfrei ist. Dieser Vorteil ist nämlich auf die Aufgabenstellung „Alternativ dürfen Sie ...“ zurückzuführen, die keinen „Rechenweg“ verlangt. Sollte die Lösung doch ein paar konkrete Teilworte erwähnen, müsste sie dann als die erste Alternative betrachtet und dem oben genannten Schema entsprechend bewertet werden.

Lösung Aufgabe 6 (Kellerautomat): **(15 Punkte)**

- 3 P: einfache Ausdrücke wir [2+(2+2)] werden akzeptiert
- 3 P wiederholende Klammern wir (2+2)+(2+2)werden akzeptiert
- 3 P sinnvolle Kommentare und Begründungen
- 2 P. geklammerte 2, d.h. (((2))) wird akzeptieren
- 2 P : keine falschen (,+,), usw
- 2 P keine falschen Präfixe und Suffixe

Bei verschachtelten arithmetischen Ausdrücken kann die Klammerung schnell unübersichtlich werden. Aus diesem Grund wollen wir auch eckige Klammern zulassen. Ausdrücke die eckige Klammern enthalten dürfen selbst nur noch mit eckigen Klammern geklammert werden. Der Einfachheit halber lassen wir nur den binären Operator + und nur die Zahl 2 zu.

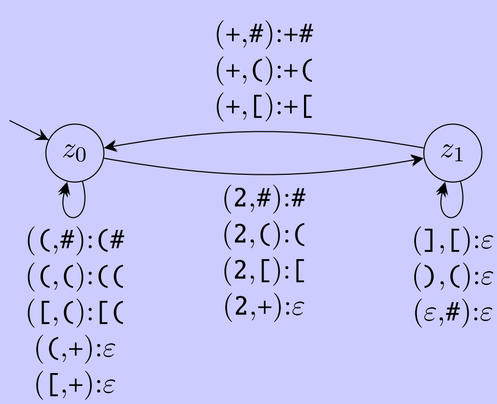
Valide Ausdrücke sind also beispielsweise $[2 + ((2 + 2) + 2) + (2 + 2)]$ und $[[2 + (2 + (2 + 2))] + 2] + 2$, aber nicht $(2 + [(2 + (2 + 2))])$, da ein Ausdruck mit eckigen Klammern in runden geklammert wurde.

Geklammerte einzelne Zweien wie ((2)) oder [[((2))]] sind erlaubt.

Zeigen Sie, dass die beschriebene Sprache der geklammerten arithmetischen Ausdrücke kontextfrei ist, indem Sie einen Kellerautomaten angeben, der diese Sprache akzeptiert.

Erläutern Sie die Konstruktion Ihres Automaten in wenigen Sätzen: Welche Zustände dienen wozu? Welche Kellersymbole dienen wozu? Warum gibt es welche Zustandsübergänge?

LÖSUNG: Eine Möglichkeit ist es, auf dem Keller, die geöffneten Klammern zu speichern D.h. die Kellersymbole sind `#`, `(`, `[` , `+`, dabei markiert `#` den Kellerboden, `(`, dass es eine geöffnete runde Klammer gibt, `[` markiert geöffnete eckige Klammern, `+` markiert, dass ein + gelesen wurde und der rechte Operand noch nicht (vollständig) eingelesen wurde. Der Automat dazu:



Die Idee dabei ist, dass im Zustand z_0 die Klammern geöffnet werden, die im Zustand z_1 geschlossen werden. In Zustand z_1 kann akzeptiert werden, wenn der Keller nur das Bodensymbol `#` enthält. Wird ein Operator gelesen, wird in z_0 gewechselt um den rechten Operanden sicher einzulesen.

Lösung Aufgabe 7 (Pumpinglemmata): **(18 Punkte)**

Punkteverteilung: a) 9 Punkte b) 9 Punkte

a) Zur Erinnerung:

Pumpinglemma für reguläre Sprachen.

Sei L eine reguläre Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$, das Mindestlänge n hat (d.h. $|z| \geq n$), als $z = uvw$ geschrieben werden kann, so dass gilt: i) $|uv| \leq n$, ii) $|v| \geq 1$ und iii) für alle $i \geq 0$: $uv^iw \in L$.

Zeigen Sie, dass die Sprache $L = \{w \in \{a,b\}^* \mid \#_a(w) < \#_b(w)\}$ nicht regulär ist.

LÖSUNG: Sei $n \in \mathbb{N}_{>0}$ beliebig. Sei $z = a^n b^{n+1}$. Dann gilt $z \in L$ und $|z| \geq n$. Sei $z = uvw$ mit $|uv| \leq n$ und $|v| \geq 1$. Dann gilt $u = a^r$, $v = a^s$, $w = a^t b^{n+1}$ mit $s > 0$, $r+s+t = n$. Dann ist $uv^2w = a^m b^{n+1}$ mit $m = n + s \geq n+1$ und daher $uv^2w \notin L$. Daher erfüllt L die Pumpingeigenschaft für reguläre Sprachen nicht und das Pumpinglemma für reguläre Sprachen zeigt, dass L nicht regulär sein.

4P. geeignetes Wort gewählt (1 liegt in L, 1 $|z| \geq n$, 2 geeignet zum Pumpen)
5P korrekt rausgepumpt

b) Zur Erinnerung:

Pumpinglemma für kontextfreie Sprachen.

Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$, das Mindestlänge n hat (d.h. $|z| \geq n$), als $z = uvwxy$ geschrieben werden kann, so dass gilt: i) $|vx| \geq 1$, ii) $|vwx| \leq n$ und iii) für alle $i \geq 0$: $uv^iwx^iy \in L$.

Zeigen Sie, dass die Sprache $L = \{a^i b^{2i} c^{3i} \mid i \in \mathbb{N}\}$ nicht kontextfrei ist.

LÖSUNG: Sei $n \in \mathbb{N}_{>0}$ beliebig. Sei $z = (a^n b^{2n} c^{3n})$. Dann gilt $z \in N$ und $|z| \geq n$. Sei $z = uvwxy$ mit $|vwx| \leq n$ und $|vx| \geq 1$. Wenn $|vwx|$ Teilwort von $a^n b^{2n}$, dann enthält uv^0wx^0y genau $3n$-mal c am Ende, aber $\#_a(uv^0wx^0y) < n$ oder $\#_b(uv^0wx^0y) < 2n$ und daher $uv^0wx^0y \notin L$. Wenn $|vwx|$ Teilwort von $b^{2n}c^{3n}$, dann enthält uv^0wx^0y genau n-mal a am Anfang, aber $\#_b(uv^0wx^0y) < 2n$ oder $\#_c(uv^0wx^0y) < 3n$ und daher $uv^0wx^0y \notin L$.

4P. geeignetes Wort gewählt (1 liegt in L, 1 $|z| \geq n$, 2 geeignet zum Pumpen) 5P korrekt rausgepumpt

Lösung Aufgabe 8 ($\mathcal{NP}$-Vollständigkeit): **(12 Punkte)**

Punkteverteilung: a) 3 Punkte b) 9 Punkte

Das PARTITION-Problem kann in der gegeben/gefragt-Notation definiert werden durch:

PARTITION

gegeben: Eine Folge von natürlichen Zahlen $S = (s_1, \dots, s_n)$, wobei $\forall i : s_i \geq 1$.

gefragt: Gibt es eine Menge I von Indizes $\{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}$, sodass $\sum_{i \in I} s_i = \sum_{i \in \{1,\dots,n\} \setminus I} s_i$ erfüllt ist (d. h. I partitioniert S in zwei Partitionen deren Summen identisch sind)?

Das PARTITION-Problem ist $\mathcal{NP}$-vollständig. Das Multimaschinen-Aufgaben-Scheduling-Problem ist:

MULTIMASCHINEN-AUFGABEN-SCHEDULING (MMAS)

gegeben: Ein Tupel $((M_1, \dots, M_n), (B_1, \dots, B_m), Z)$, wobei $M_1, \dots, M_n$ eine Folge von parallel arbeitenden gleichartigen Maschinen ist, $A_1, \dots, A_m$ eine Folge von Maschinenaufgaben ist, sodass B_i die benötigte Bearbeitungszeit für Aufgabe A_i (in Sekunden) ist, und Z eine Zeitschranke in Sekunden ist. Eine Maschine kann nur eine Aufgabe gleichzeitig bearbeiten und Aufgaben können während ihrer Bearbeitung die Maschine nicht wechseln.

gefragt: Gibt es eine Zuordnung $\mathcal{M} : \{1, \dots, m\} \to \{1, \dots, n\}$, die jede Aufgabe A_k zu einer Maschine $M_{\mathcal{M}(k)}$ zuweist, welche die Aufgabe A_k in B_k Sekunden bearbeitet, sodass alle Aufgaben in nicht mehr als Z Sekunden ausgeführt werden können?

a) Zeigen Sie, dass die MMAS-Instanz

$$((M_1, M_2, M_3), (B_1, B_2, B_3, B_4, B_5), Z) = ((M_1, M_2, M_3), (15, 30, 25, 40, 50), 60)$$

lösbar ist, indem Sie die entsprechende Zuordnung $\mathcal{M} : \{1, 2, 3, 4, 5\} \to \{1, 2, 3\}$ angeben und prüfen, dass diese die Zeitschranke von 60 Sekunden einhält.

LÖSUNG: Um die Zeitschranke von 60 Sekunden einzuhalten können auf jeder Maschine maximal 2 Aufgaben durchgeführt werden. Aufgabe 5 mit 50 Sekunden kann mit keiner anderen kombiniert werden ohne das Zeitlimit zu überschreiten. Aufgabe 4 mit 40 Sekunden kann nur mit Aufgabe 1 kombiniert werden. Daher muss Aufgabe 2 mit Aufgabe 3 kombiniert werden.

Daher sind mögliche Lösungen von der Form

$$\begin{aligned} \mathcal{M}(1) &= i \\ \mathcal{M}(2) &= j \\ \mathcal{M}(3) &= j \\ \mathcal{M}(4) &= i \\ \mathcal{M}(5) &= k \end{aligned}$$

mit $i, j, k \in \{1, 2, 3\}$ und $i \neq j, i \neq k, j \neq k$.

nur 1 P wenn die Aufgaben die Maschinen wechseln.

b) Zeigen Sie, dass MMAS $\mathcal{NP}$-vollständig ist. Führen Sie dafür unter anderem eine Polynomialzeitreduktion von PARTITION auf MMAS durch.

LÖSUNG: Für den $\mathcal{NP}$-Vollständigkeitsnachweis ist zu zeigen, dass MMAS $\in \mathcal{NP}$ gilt und dass MMAS $\mathcal{NP}$-schwer ist.

- MMAS $\in \mathcal{NP}$: Eine nichtdeterministische Turingmaschine liest die (Kodierung) der MMAS-Instanz und rät nichtdeterministisch für jede Aufgabe, die Maschine auf der sie durchgeführt wird. Anschließend summiert die Turingmaschine die Ausführungszeit pro Maschine und berechnet schließlich das Maximum. Wenn das Maximum nicht größer als die Schranke Z ist, dann akzeptiert die Turingmaschine auf diesem Pfad. Die Berechungen sind in Polynomialzeit durchführbar. Daher ist MMAS in Polynomialzeit auf einer nichtdeterministischen Turingmaschine entscheidbar.
- MMAS ist $\mathcal{NP}$-schwer: Sei f die folgende Funktion, die jede PARTITION-Instanz $S = (s_1, \ldots, s_n)$ in eine MMAS-Instanz überführt: $f(S) = ((M_1, M_2), (s_1, \ldots, s_n), \lfloor \frac{\sum_{i=1,\ldots,n} s_i}{2} \rfloor)$. Die Funktion f ist offensichtlich total und in Polynomialzeit auf einer deterministischen Turingmaschine berechenbar.

  Es gilt: PARTITION-Instanz $S = \{s_1, \ldots, s_n\}$ hat eine Lösung genau dann, wenn $f(S)$ eine MMAS-Lösung hat:

„$\Rightarrow$“ Sei $I = \{i_1, \ldots, i_k\}$ eine Indexmenge mit $\sum_{i \in I} s_i = \sum_{i \in \{1,\ldots,n\} \setminus I} s_i$. Dann muss gelten $\sum_{i \in I} s_i = \lfloor \sum_{i=1}^{n} s_i/2 \rfloor = \sum_{i=1}^{n} s_i/2$ und daher gilt für die Zuordnung

$$\mathcal{M}(i) = \begin{cases} 1, \text{wenn } i \in I \\ 2, \text{wenn } i \notin I \end{cases}$$

dass sie die Maschinenaufgaben derart auf die Maschinen M_1 und M_2 verteilt, dass insgesamt nicht mehr als $\lfloor \sum_{i=1}^{n} s_i/2 \rfloor$ Zeit verbraucht wird.

„$\Leftarrow$“ Sei $\mathcal{M} : \{1, \ldots, n\} \to \{1, 2\}$ eine Zuordnung der Aufgaben auf die Maschinen M_1 und M_2, sodass die Laufzeit beider Maschinen nicht mehr als $\lfloor \frac{\sum_{i=1}^{n} s_i}{2} \rfloor$ Sekunden verbraucht. Dann muss gelten, dass beide Maschinen genau $\frac{\sum_{i=1}^{n} s_i}{2}$ Sekunden laufen (anders lassen sich die $\sum_{i=1}^{n} s_i$ Sekunden nicht aufteilen). Daher ist $I := \{i \mid \mathcal{M}(i) = 1\}$ eine Indexmenge, die die PARTITION-Instanz löst.

in NP: 2 Punkte, NP-hart: 7 Punkte, wobei 5 Punkte für die richtige Kodierung, 1 Punkt für Korrektheitsbeweis, 1 Punkt für Beweis Kodierung polynomiell berechenbar und total.

1 Punkt wenn nur das Schema da ist.