1. Klausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2014/2015

Lösung!

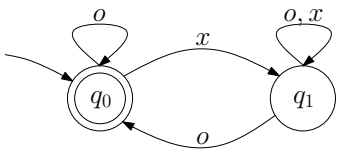
Beachten Sie:

- Bringen Sie den Aufkleber mit Ihrem Namen und Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

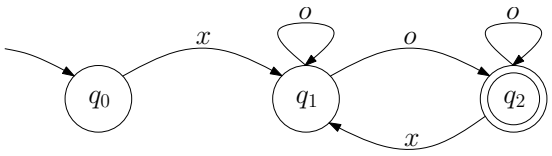
Aufgabe	Mögliche Punkte						Erreichte Punkte					
	a	b	c	d	e	Σ	a	b	c	d	e	Σ
1	2	2	3	–	–	7				–	–	
2	2	2	2	–	–	6				–	–	
3	2	4	2	–	–	8				–	–	
4	1	3	1	–	–	5				–	–	
5	1	1	1	3	2	8						
6	1	3	3	1	–	8					–	
7	5	3	–	–	–	8			–	–	–	
8	4	–	–	–	–	4		–	–	–	–	
9	6×1					6						
Σ						60						

Problem 1: Endliche Automaten 2+2+3=7 Punkte

Gegeben seien die endlichen Automaten $\mathcal{A}_1 = (Q = \{q_0, q_1\}, \Sigma = \{o, x\}, \delta_1, q_0, \{q_0\})$ und $\mathcal{A}_2 = (Q = \{q_0, q_1, q_2\}, \Sigma = \{o, x\}, \delta_2, q_0, \{q_2\})$, wobei δ_1 und δ_2 durch folgende Zustandsdiagramme gegeben sind. Der Fehlerzustand ist implizit gegeben.



Endlicher Automat $\mathcal{A}_1$ mit δ_1



Endlicher Automat $\mathcal{A}_2$ mit δ_2

(a) Geben Sie jeweils einen regulären Ausdruck für die Sprache an, die von $\mathcal{A}_1$ bzw. $\mathcal{A}_2$ akzeptiert wird. *Hinweis:* Hier ist nicht verlangt, dass Sie das Verfahren aus der Vorlesung verwenden.

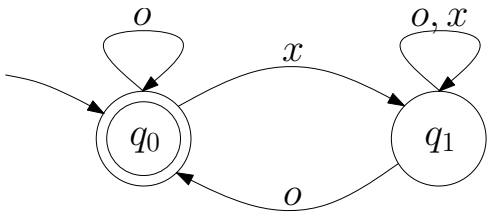
Lösung:

$\mathcal{A}_1$: $(o^\star x(x \cup o)^\star o)^\star \cup o^\star$

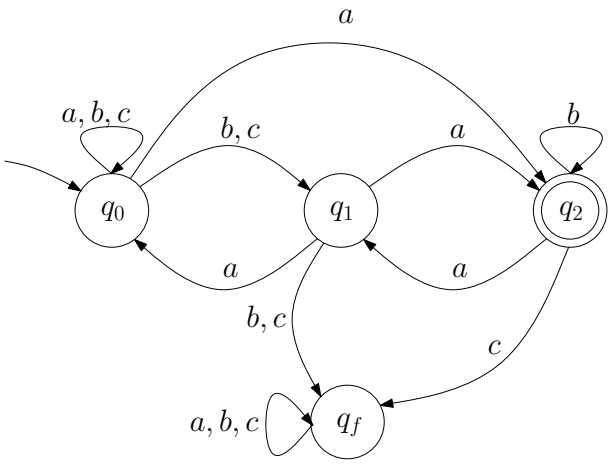
$\mathcal{A}_2$: $x \underbrace{o^\star o o^\star}_{o^+}(x \underbrace{o^\star o o^\star}_{o^+})^\star \quad \Leftrightarrow \quad xo^+(xo^+)^\star \quad \Leftrightarrow \quad (xo^+)^+$

(b) Konstruieren Sie einen endlichen Automaten mit maximal 5 Zuständen, der die Sprache $L = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ akzeptiert.

Lösung:



(c) Gegeben sei der endliche Automat $\mathcal{A}_3 = (Q = \{q_0, q_1, q_2, q_f\}, \Sigma = \{a, b, c\}, \delta_3, q_0, \{q_2\})$, wobei δ_3 durch folgendes Zustandsdiagramm gegeben ist.



Geben Sie zu $\mathcal{A}_3$ einen äquivalenten vollständigen deterministischen endlichen Automaten tabellarisch an und verwenden Sie hierfür die Potenzmengenkonstruktion.

Lösung:

Durchführung der Potenzmengenkonstruktion:

Zustand	Übergänge		
	a	b	c
$\{q_0\}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$
$\underline{\{q_0, q_2\}}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_f\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_f\}$	$\{q_0, q_1, q_f\}$
$\underline{\{q_0, q_1, q_2\}}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_f\}$	$\{q_0, q_1, q_f\}$
$\{q_0, q_1, q_f\}$	$\{q_0, q_2, q_f\}$	$\{q_0, q_1, q_f\}$	$\{q_0, q_1, q_f\}$
$\underline{\{q_0, q_1, q_2, q_f\}}$	$\{q_0, q_1, q_2, q_f\}$	$\{q_0, q_1, q_2, q_f\}$	$\{q_0, q_1, q_f\}$
$\underline{\{q_0, q_2, q_f\}}$	$\{q_0, q_1, q_2, q_f\}$	$\{q_0, q_1, q_2, q_f\}$	$\{q_0, q_1, q_f\}$

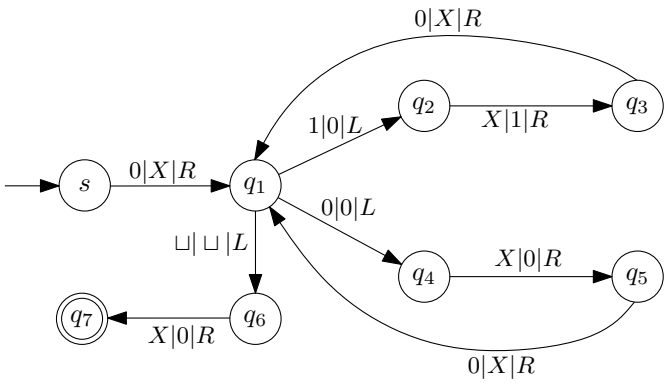
Tabelle 1: Potenzmengenkonstruktion von $\mathcal{A}_3$.

Problem 2: Turingmaschine 2+2+2=6 Punkte

Gegeben sei die folgende Turingmaschine

$$\mathcal{M} = (Q = \{s, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \Sigma = \{0, 1\}, \Gamma = \Sigma \cup \{\sqcup, X\}, \delta, s, F = \{q_7\}).$$

Die Übergangsfunktion δ von $\mathcal{M}$ ist durch das untenstehende Zustandsübergangsdiagramm dargestellt.



Für $\mathcal{M}$ gelten folgende **Konventionen**: $\mathcal{M}$ hält und akzeptiert sobald ein Endzustand erreicht wird. Alle Übergänge, die aus Endzuständen herausführen, werden demnach nicht verwendet. $\mathcal{M}$ hält und **akzeptiert nicht** für nicht dargestellte Zustandsübergänge.

(a) Dokumentieren Sie die Berechnung des Wortes 01101. Geben Sie dazu für **jeden** Schritt die aktuelle Konfiguration an.

Lösung:

$(s)01101$
$X(q_1)1101$
$(q_2)X0101$
$1(q_3)0101$
$1X(q_1)101$
$1(q_2)X001$
$11(q_3)001$
$11X(q_1)01$
$11(q_4)X01$
$110(q_5)01$
$110X(q_1)1$
$110(q_2)X0$
$1101(q_3)0$
$1101X(q_1)\sqcup$
$1101(q_6)X\sqcup$
$11010(q_7)\sqcup$

(b) Geben Sie die Sprache $L(\mathcal{M}) \subseteq \Sigma^\star$ an, die $\mathcal{M}$ akzeptiert. Geben Sie außerdem die Funktion $f_\mathcal{M} \colon \Sigma^\star \to \Gamma^\star$ an, die $\mathcal{M}$ realisiert. Begründen Sie Ihre Antwort, indem Sie das Verhalten der Turingmaschine auf einer Eingabe informell beschreiben.

Lösung: $L(\mathcal{M}) = \{0w \mid w \in \Sigma^\star\}$

Sei $w = x_1 x_2 \dots x_n$ mit $x_i \in \Sigma$ ein beliebiges Wort aus $\Sigma^\star$. Die realisierte Funktion von $\mathcal{M}$ ist dann

$$f_\mathcal{M}(w) = \begin{cases} x_2 \dots x_n x_1 & \text{falls } x_1 = 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

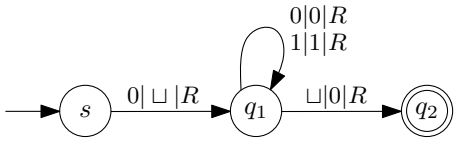
Anfangs steht der Lesekopf ganz links. Falls das erste Zeichen eine 0 ist, so beginnt $\mathcal{M}$ die Abarbeitung. Andernfalls ist das Verhalten von $\mathcal{M}$ undefiniert. Sei also das erste Zeichen eine 0. $\mathcal{M}$ ersetzt die 0 durch ein X und geht ein Schritt nach rechts und befindet sich nun im Zustand q_1. Liest $\mathcal{M}$ nun ein $\sqcup$, ersetzt $\mathcal{M}$ das X mit einer 0 und akzeptiert. Liest dagegen $\mathcal{M}$ eine 1 oder eine 0, gerät $\mathcal{M}$ in die Schleife $q_1 q_2 q_3 q_1$ bzw. in die Schleife $q_1 q_4 q_5 q_1$. Beide Schleifen verhalten sich ähnlich: In der ersten Schleife wird die 1 durch eine 0 ersetzt und das X um eine Stelle nach rechts verschoben und in der zweiten Schleife wird die 0 durch eine 0 ersetzt und das X um eine Stelle nach rechts verschoben. Die Schleifen propagieren also die 0, die am Anfang gelesen wurde, durch das Wort durch.

(c) Geben Sie eine Turingmaschine $\mathcal{M}'$ an, die folgende Eigenschaften besitzt.

- $\mathcal{M}'$ akzeptiert genau die Sprache $L(\mathcal{M})$.
- $\mathcal{M}'$ realisiert dieselbe Funktion wie $\mathcal{M}$.
- $\mathcal{M}'$ hat maximal drei Zustände.

Hinweis: Sie dürfen dieselben Konventionen verwenden, wie sie für $\mathcal{M}$ angegeben sind.

Lösung:



Problem 3: Reguläre Sprachen 2+4+2=8 Punkte

(a) Zeigen oder widerlegen Sie, dass die Sprache $L_1 = \{a^n b^k a^n \in \{a,b\}^\star \mid k \in \mathbb{N}_0, n \in \mathbb{N} \setminus \{0\}, n \leq 3\}$ regulär ist.

Lösung: **L_1 ist regulär.**
Man kann zeigen, dass der reguläre Ausdruck $\{ab^\star a \mid aab^\star aa \mid aaab^\star aaa\}$ alle von L_1 erzeugten Sprachen akzeptiert. Alternativ kann man einen zu L_1 gehörigen DEA konstruieren.

(b) Zeigen oder widerlegen Sie, dass die Sprache $L_2 = \{a^{i!} \in \{a\}^\star \mid i \in \mathbb{N}\}$ regulär ist.

Lösung: **L_2 ist nicht regulär:**

Beweis: Annahme L_2 ist regulär, dann müsste das Pumping Lemma gelten. Es existiert also eine Zahl $n \in \mathbb{N}$, sodass für das Wort $w = a^{n!}$ die Zerlegung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$ existiert **und** $uv^i x \in L_2$ für alle $i \in \mathbb{N}_0$.

Betrachte die Zerlegung $u = a^p$, $v = a^q$ und $x = a^r$ mit $p+q+r = n!$, $p+q \leq n$ und $1 \leq q \leq n$. Wobei $n! = 1 \cdot 2 \cdot \cdots \cdot (n-1) \cdot n$. Sei nun die Zerlegung für p, q und r wie folgt

$$w = uvx = a^{n!} = \overbrace{\underbrace{aaaaaa\ldots}_{u}}^{p=n-q}\overbrace{\underbrace{aaaaaa\ldots}_{v}}^{q}\overbrace{\underbrace{aaaaaa\ldots}_{x}}^{r=n!-n}.$$

Pumpen wir nun an der Stelle v, so muss laut Annahme $uv^2x \in L_2$ gelten. Dabei gilt, dass $p + 2q + r = n! + q$. Es gilt

$$n! < n! + q \leq n! + n \leq n! \cdot n < n! \cdot (n+1) = (n+1)!.$$

Damit gilt, dass es keine Zerlegung uvx für L_2 gibt, sodass $uv^2x \in L_2$.

(c) Zeigen Sie mithilfe eines Gegenbeispiels, dass die Umkehrung des Pumping-Lemmas nicht gilt. Begründen Sie kurz Ihre Antwort.

Lösung: Eine Sprache bei der das Pumping-Lemma für reguläre Sprachen scheitert, ist $L = \{w \in \{0,1\}^\star \mid w = 1^i, i \geq 0 \text{ oder } w = 0^j 1^{i^2}, i,j \geq 1\}$.

- Falls $w = 1^i$, so ist auch $uv^i x$ vom Typ $1^\ell \in L_3$.
- Falls $w = 0^j 1^{i^2}$, so ist auch $uv^0 x \in L_3$. Für $j = 1$ ist $uv^0 x = x = 1^{i^2}$. Für $k \geq 1$ gilt $uv^k x = 0^{j+k} 1^{i^2} \in L_3$.

Problem 4: Berechenbarkeit – Halteproblem 1+3+1=5 Punkte

(a) Sei $L_1 = \{\langle \mathcal{M} \rangle \mid$ Die Turingmaschine $\mathcal{M}$ berechnet das Spiegelwort der Eingabe$\}$. Zeigen oder widerlegen Sie, dass L_1 entscheidbar ist.

Lösung: Nach Satz von Rice ist die Sprache nicht entscheidbar.

(b) Gegeben sei die folgende Sprache über dem Alphabet $\Sigma = \{0, 1\}$

$$L_2 = \{(v, w) \in \Sigma^\star \times \Sigma^\star \mid \text{Die Turingmaschinen } T_v \text{ und } T_w \text{ halten auf exakt denselben Eingaben}\}.$$

Dabei bezeichnet T_w die Turingmaschine, die durch die Gödelnummer w codiert ist. Zeigen Sie, dass L_2 nicht entscheidbar ist.
Hinweis: Sie dürfen verwenden, dass die Sprache $L_3 = \{w \mid T_w$ hält auf keiner Eingabe$\}$ nicht entscheidbar ist.

Lösung: Annahme es gibt eine Turingmaschine $\mathcal{M}$, die L_2 entscheidet. Wir zeigen, dass dann auch L_3 entscheidbar wäre, indem wir eine Turingmaschine $\mathcal{M}'$ konstruieren, die L_3 entscheidet, d.h. für alle Eingaben $v \in \Sigma^\star$ liefert $\mathcal{M}'$

$$\mathcal{M}'(v) = \begin{cases} \text{Ja} & T_v \text{ hält auf keiner Eingabe.} \\ \text{Nein} & \textit{sonst} \end{cases}$$

Für $\mathcal{M}$ und alle Eingaben $v, w \in \Sigma^\star$ gilt

$$\mathcal{M}(v, w) = \begin{cases} \text{Ja} & T_v \text{ und } T_w \text{ halten auf exakt denselben Eingaben} \\ \text{Nein} & \textit{sonst} \end{cases}$$

Sei $\mathcal{M}''$ eine Turingmaschine, die niemals hält, zum Beispiel eine einfache Endlosschleife. Damit gilt $\langle M \rangle \in L_3$. Dann definieren wir $\mathcal{M}'$ als

$$\mathcal{M}'(v) := \mathcal{M}(v, \langle \mathcal{M}'' \rangle)$$

Da $\mathcal{M}''$ auf keiner Eingabe hält, gilt nach Definition von L_2, dass $v \in L_3$ falls $\mathcal{M}$ akzeptiert und sonst $v \notin L_3$.

(c) Sei L_4 eine unentscheidbare Sprache. Zeigen oder widerlegen Sie folgende Aussage: Jede Teilmenge von L_4 ist ebenfalls unentscheidbar.

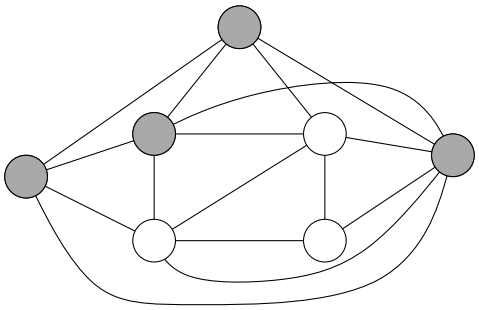
Lösung: Die Aussage gilt nicht. Sei $\Sigma = \{0, 1\}$. Das Halteproblem $H \subseteq \Sigma^\star$ ist nicht entscheidbar. Allerdings ist die Sprache $\emptyset \subseteq H$ entscheidbar.

Problem 5: CLIQUE – NP-Vollständigkeit 1+1+1+3+2=8 Punkte

Sei $G = (V, E)$ ein ungerichteter Graph. Die Menge $C \subseteq V$ heißt *Clique* von G, falls für alle Knoten $u, v \in C$ mit $u \neq v$ gilt $\{u, v\} \in E$. Das NP-vollständige Problem CLIQUE ist dann wie folgt definiert:

Problem CLIQUE
Gegeben: Ungerichteter Graph $G = (V, E)$ und Parameter k.
Frage: Existiert eine Clique C von G mit $|C| \geq k$?

(a) Existiert eine Clique C mit $|C| \geq 4$ in dem unten abgebildeten Graphen? Falls ja, dann markieren Sie die Knoten einer solchen Clique. Andernfalls begründen Sie Ihre Antwort.



(b) Geben Sie für CLIQUE das entsprechende Optimierungs- und Optimalwertproblem an. *Lösung:*

Optimalwertproblem:
Gegeben: Ungerichteter Graph $G = (V, E)$.
Gesucht: Größte mögliche Zahl $k \in \mathbb{N}$, sodass es Clique $V' \subseteq V$ mit $|V'| \geq k$ gibt.

Optimierungsproblem:
Gegeben: Ungerichteter Graph $G = (V, E)$.
Gesucht: Größte Clique $V' \subseteq \mathcal{V}$ in G.

(c) Zeigen Sie, dass wenn CLIQUE in der Komplexitätsklasse P liegt, dann gilt P = NP.

Lösung: Da CLIQUE ein NP-vollständiges Problem ist, gibt es für jedes Problem $\Pi \in \mathsf{NP}$ eine polynomielle Transformation $\propto$ mit $\Pi \propto$ CLIQUE. Ein Problem $\Pi \in \mathsf{NP}$ kann also in polynomieller Zeit gelöst werden, indem man Π zuerst auf CLIQUE reduziert und dann CLIQUE in polynomieller Zeit löst.

(d) Das Problem GANZZAHLIGESPROGRAMM ist wie folgt definiert.

Problem GANZZAHLIGESPROGRAMM
Gegeben: $a_{ij} \in \mathbb{Z}_0$, $b_i, c_j \in \mathbb{Z}_0$, $1 \leq i \leq m$, $1 \leq j \leq n$, $B \in \mathbb{Z}_0$
Frage: Existieren $x_1, \ldots, x_n \in \mathbb{N}_0$, sodass

$$\sum_{j=1}^{n} c_j \cdot x_j = B \quad \text{und} \quad \sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i \quad \text{für } 1 \leq i \leq m?$$

Geben Sie eine polynomielle Reduktion von dem Entscheidungsproblem CLIQUE auf GANZZAHLIGESPROGRAMM an. Geben Sie insbesondere die Interpretation der eingeführten Variablen an und begründen Sie die Korrektheit der Reduktion.

Lösung: Führe für jeden Knoten $u \in V$ eine binäre Variable x_u ein, die bestimmt, ob u in der gesuchten Clique enthalten ist. Die Reduktion ist polynomiell, da die Anzahl Variablen und Nebenbedingungen in $O(|V|^2)$ liegen.

Nebenbedingungen:

- Für jeden Knoten $u \in V$: $x_u \leq 1$ und $-x_u \leq 0$. Diese Nebenbedingungen stellen sicher, dass $x_u \in \{0, 1\}$.
- Für jedes Knotenpaar $u, v \in V$ mit $u \neq v$ und $\{u, v\} \notin E$ die Nebenbedingung $x_u + x_v \leq 1$. Stellt sicher, dass nicht adjazente Knoten nicht gemeinsam in der Clique sein können.
- $\sum_{u \in V} x_u \geq k$. Stellt sicher, dass mindestens k Knoten zur Clique gehören.

(e) Welches in der Vorlesung vorgestellte Entscheidungproblem modelliert das unten stehende ganzzahlige Programm für einen gegeben Graphen $G = (V, E)$? Begründen Sie Ihre Antwort.

Ganzzahliges Programm WASMAGDASWOHLSEIN:

Für jeden Knoten $u \in V$ gibt es die Variablen x_a^u, x_b^u und x_c^u. Außerdem sollen folgende Ungleichungen gelten.

$$\begin{array}{rlll}
\text{Für alle } u \in V: & x_a^u + x_b^u + x_c^u \leq 1 & -x_a^u - x_b^u - x_c^u \leq -1 & (1) \\
\text{Für alle } \{u, v\} \in E: & x_a^u + x_a^v \leq 1 & x_b^u + x_b^v \leq 1 \quad x_c^u + x_c^v \leq 1 & (2) \\
\text{Für alle } u \in V: & x_a^u \leq 1 & x_b^u \leq 1 \quad x_c^u \leq 1 & (3) \\
\text{Für alle } u \in V: & -x_a^u \leq 0 & -x_b^u \leq 0 \quad -x_c^u \leq 0 & (4)
\end{array}$$

Lösung: Das ganzzahlige Programm modelliert das Problem 3COLOR. Für jeden Knoten $u \in V$ gibt es eine Variable, die einer Farbe a, b oder c entspricht. Zum Beispiel bedeutet $x_a^u = 1$, dass der Knoten u die Farbe a besitzt. Die Bedingungen vom Typ (1) stellen sicher, dass jeder Knoten nur eine Farbe hat. Die Bedingungen vom Typ (2) stellen sicher, dass adjazente Knoten nicht eine gemeinsame Farbe besitzen. Die Bedingungen vom Typ (3) und (4) sorgen dafür dass für jede Variabel x_a^u gilt, dass $x_a^u \in \{0, 1\}$ (analog x_b^u und x_c^u).

Problem 6: HittingSet – Approximation 1+3+3+1=8 Punkte

Das Problem 4-HittingSet ist wie folgt definiert:

Problem 4-HittingSet

Gegeben: Universum $\mathcal{U} = \{1, \ldots, n\}$ und Mengensystem $\mathcal{C} \subseteq 2^{\mathcal{U}}$, wobei $2^{\mathcal{U}}$ die Potenzmenge von $\mathcal{U}$ bezeichnet. Jede Menge $A \in \mathcal{C}$ enthalte maximal 4 Elemente, d.h. $|A| \leq 4$.

Gesucht: Hitting Set X für $\mathcal{C}$ mit minimaler Kardinalität.

Hinweis: Eine Menge $X \subseteq \mathcal{U}$ heißt *Hitting Set* für $\mathcal{C}$, falls für jedes $A \in \mathcal{C}$ gilt $A \cap X \neq \emptyset$.

Betrachten Sie folgenden Algorithmus:

Algorithmus 1 : HittingSetApproximation

Eingabe : $\mathcal{U} = \{1, \ldots, n\}$, $\mathcal{C} \subseteq 2^{\mathcal{U}}$ mit $|A| \leq 4$ für alle $A \in \mathcal{C}$.
Ausgabe : Hitting Set X für $\mathcal{C}$.

```
1 X ← ∅
2 solange C ≠ ∅ tue
3 |   A ← wähle eine beliebige Menge A ∈ C.
4 |   X ← X ∪ A
5 |   C ← C \ {A ∈ C | A ∩ X ≠ ∅}
```

Für eine beliebige Instanz I von 4-HittingSet, bezeichne $\mathcal{A}(I)$ die Kardinalität des von Algorithmus 1 zurückgegebenen Hitting Sets, sowie $\mathrm{OPT}(I)$ die Kardinalität einer optimalen Lösung für I.

(a) Geben Sie die Menge X an, die Algorithmus 1 bei Eingabe der folgenden Instanz berechnet:

$$\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7\}, \qquad \mathcal{C} = \{c_1 = \{2, 3, 6\}, c_2 = \{1, 3, 4, 7\}, c_3 = \{4, 6, 7\}, c_4 = \{1, 5\}\}.$$

In Zeile 3 werde jeweils die Menge c_i mit dem kleinsten Index i unter allen Mengen in $\mathcal{C}$ gewählt.

Lösung: $X = \{1, 2, 3, 5, 6\}$

(b) Geben Sie eine Familie $I_1 = (\mathcal{U}_1, \mathcal{C}_1), I_2 = (\mathcal{U}_2, \mathcal{C}_2), I_3 = (\mathcal{U}_3, \mathcal{C}_3), \ldots$ von Instanzen für 4-HittingSet an, sodass gilt $|\mathcal{U}_k| \in \Omega(k)$, $|\mathcal{C}_k| \in \Omega(k)$ und $\mathcal{A}(I_k) = 4\,\mathrm{OPT}(I_k)$ für alle $k \in \mathbb{N}$. Begründen Sie Ihre Antwort.

Lösung:

I_1: $\mathcal{U}_1 = \{1, 2, 3, 4\}$, $\mathcal{C}_1 = \{\{1, 2, 3, 4\}\}$
I_2: $\mathcal{U}_2 = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $\mathcal{C}_2 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$
...
I_k: $\mathcal{U}_k = \{1, \ldots, 4k\}$, $\mathcal{C}_k = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \ldots, \{4k-3, 4k-2, 4k-1, 4k\}\}$

Da die Mengen in $\mathcal{C}_k$ paarweise disjunkt sind, enthält ein optimale Lösung $X^\star$ jeweils ein Element jeder Menge. Es gilt also $\mathrm{OPT}(I) = |X^\star| = |\mathcal{U}|/4$. Dagegen fügt Algorithmus 1 in jedem Schleifendurchlauf eine Menge $A \in \mathcal{C}$ zu X hinzu. Da die Mengen in $\mathcal{C}_k$ paarweise disjunkt sind, wird ausschließlich A aus $\mathcal{C}_k$ entfernt. Am Schluss gilt also $\mathcal{A}(I_k) = |X| = |\mathcal{U}|$. Folglich gilt $\mathcal{A}(I) = 4\,\mathrm{OPT}(I)$.

(c) Betrachten Sie eine (feste) optimale Lösung $X^\star$. Zeigen Sie, dass jede in Zeile 3 ausgewählte Menge A mindestens ein Element $z \in X^\star \setminus X$ enthält.

Lösung: Da $X^\star$ ein Hitting Set ist, gilt nach Definition $X^\star \cap A \neq \emptyset$. Sei also $z \in X^\star \cap A$ beliebig, aber fest. Wir nehmen nun an, dass $z \in X$ gilt. Dann wurde die Schleife mindestens einmal durchlaufen, da X als leere Menge initialisiert wurde. Dies ist aber ein Widerspruch, da in diesem Fall A in Zeile 5 aus $\mathcal{C}$ entfernt worden wäre. Also ist $z \in A \cap (X^\star \setminus X)$ wie behauptet.

(d) Zeigen Sie, dass für jede beliebige Instanz I von 4-HITTINGSET gilt $\mathcal{A}(I) \leq 4\,\mathrm{OPT}(I)$.

Lösung: In jedem Durchlauf von Zeile 4 werden wegen $|A| \leq 4$ höchstens vier Elemente zu X hinzugefügt, von denen mindestens eines in der optimalen Lösung $X^\star$, nicht aber in X enthalten ist. Für jedes Element $x \in X^\star$ fügen wir also höchstens einmal je Durchlauf vier Elemente zu X hinzu. Daher gilt $\mathcal{A}(I) = |X| \leq 4|X^\star| = 4 \cdot \mathrm{OPT}(I)$

Problem 7: Grammatiken und Chomsky-Hierarchien 5+3=8 Punkte

Gegeben sei ein Konzept einer imperativen Programmiersprache namens PABA. Aus dieser Programmiersprache ist folgender Auszug gegeben.

```
1  i = 30.
2
3  DO 3 TIMES.
4      i = i + 1.
5  ENDDO.
```

Ein PABA-Programm ist eine Folge von *Anweisungen*. Jede Anweisung endet mit einem '.'. PABA unterscheidet verschiedene Anweisungen.

Eine *Zuweisung* ist eine Anweisung, siehe Zeile 1 und Zeile 4. Sie besteht aus einer linken und einer rechten Seite, die durch ein '=' getrennt sind. Die linke Seite einer Zuweisung ist eine *Variable*. Die rechte Seite einer Zuweisung ist ein *Term*. Ein Term ist eine Variable, *Zahl* oder eine *Addition* zweier Terme, siehe Zeile 4. Eine Zahl ist nicht negativ. Eine Variable beginnt immer mit einem *Buchstaben* gefolgt von beliebig vielen Buchstaben und Zahlen. Eine Addition besteht aus zwei Termen getrennt durch das Symbol '+'.

Eine *do-Anweisung* beginnt mit dem Schlüsselwort `DO` gefolgt von einer *Bedingung*, siehe Zeile 3. Eine *Bedingung* besteht aus einer Variablen oder Zahl gefolgt vom Schlüsselwort `TIMES`.

Der *do-Block* ist eine Anweisung, die aus einer Folge von Anweisungen besteht, siehe Zeile 3–5: Zuerst kommt eine do-Anweisung gefolgt von beliebig vielen Anweisungen. Ein do-Block endet immer mit dem Schlüsselwort `ENDDO`.

Hinweis: Für die bessere Lesbarkeit enthält das oben angegebene Beispielprogramm Leerzeichen und Zeilenumbrüche. Diese sind nicht Teil der Grammatik.

(a) Entwerfen Sie eine Grammatik, welche die Sprache aller möglichen PABA-Programme erzeugt. Vervollständigen Sie hierzu die folgende Grammatik $G = (\Sigma, V, S, R)$.

Die Produktionen von G sind bereits teilweise unten gegeben. Die Nichtterminale aus V haben die Form $\langle Name \rangle$. Alle anderen Symbole in der folgenden Regelmenge sind Terminale, wobei ε das leere Wort bezeichnet. Wenn nötig, führen Sie weitere Terminale ein.

$$
\begin{array}{lcl}
S & \to & \langle Anweisung \rangle \\
\langle Anweisung \rangle & \to & \langle Anweisung \rangle \ \langle Anweisung \rangle \mid \langle Zuweisung \rangle \mid \langle do - Block \rangle \mid \varepsilon \\
\langle do - Block \rangle & \to & \langle do - Anweisung \rangle \ \langle Anweisung \rangle \ \texttt{ENDDO.} \\
\langle do - Anweisung \rangle & \to & \texttt{DO} \ \langle Bedingung \rangle. \\
\langle Bedingung \rangle & \to & \langle Variable \rangle \ \texttt{TIMES} \mid \langle Zahl \rangle \ \texttt{TIMES} \\
\langle Zuweisung \rangle & \to & \langle Variable \rangle = \langle Term \rangle. \\
\langle Term \rangle & \to & \langle Term \rangle + \langle Term \rangle \mid \langle Zahl \rangle \mid \langle Variable \rangle \\
\langle Variable \rangle & \to & \langle Buchstabe \rangle \langle Hilfs\,Variable \rangle \\
\langle Hilfs\,Variable \rangle & \to & \langle Hilfs\,Variable \rangle \langle Hilfs\,Variable \rangle \mid \langle Zahl \rangle \mid \langle Buchstabe \rangle \\
\langle Zahl \rangle & \to & \langle Ziffer \rangle \langle Zahl \rangle \mid \langle Ziffer \rangle \\
\langle Ziffer \rangle & \to & 0 \mid 1 \mid 2 \mid \cdots \mid 9 \\
\langle Buchstabe \rangle & \to & a \mid \cdots \mid z \mid A \mid \cdots \mid Z
\end{array}
$$

(b) Vervollständigen Sie die unten angegebene Tabelle, indem Sie für jeden Chomsky-Typ k die Sprachklasse angeben, die genau die Sprachen enthält, die von Chomsky-k-Grammatiken erzeugt werden. Geben Sie zudem für jede Sprachklasse ein Rechnermodell an, das die von Chomsky-k-Grammatiken, aber nicht die von Chomsky-$(k-1)$-Grammatiken erzeugten Sprachen akzeptiert. Geben Sie außerdem jeweils die Entscheidbarkeit des Wortproblems an.

Chomsky-Typ	**Sprachklasse**	**Rechnermodell**	**Entscheidbarkeit des Wortproblems**
Typ-0	Rekursiv aufzählbar	beliebige Turingmaschine	nein
Typ-1	Kontextsensitiv	nichtdeterministische Turinmaschine mit Speicherbedarf, der „im wesentlichen“ nicht die Länge der Eingabe überschreitet	ja
Typ-2	Kontextfrei	nichtdeterministischer Kellerautomat	ja
Typ-3	Regulär	Endlicher Automat	ja

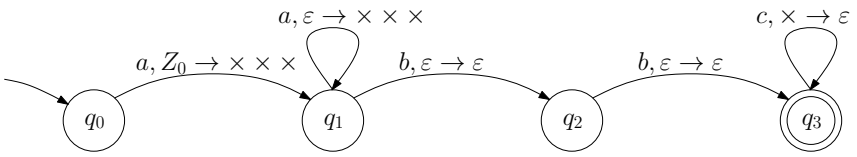
Problem 8: Kellerautomaten 4 Punkte

Gegeben seien die folgenden zwei Sprachen:

$$L_1 = \{a^n b^2 c^{3n} \in \{a,b,c\}^\star \mid n \in \mathbb{N} \setminus \{0\}\},$$
$$L_2 = \{a^n b^{2n} c^{3n} \in \{a,b,c\}^\star \mid n \in \mathbb{N}\}.$$

Geben Sie für jede der beiden Sprachen an, ob ein Kellerautomat existiert, der die Sprache erkennt. Sollte ein solcher Kellerautomat existieren, so geben Sie einen an, der maximal 4 Zustände besitzt und mit leerem Stack akzeptiert. Andernfalls geben Sie den maximalen Chomsky-Typ einer Grammatik an, welche die Sprache erzeugt.

Lösung: Sprache L_1:



Sprache L_2: Die Sprache L_2 ist nicht kontextfrei, aber kontextsensitiv: Typ 1.

Problem 9: Verschiedenes 6 Punkte

Jeder der folgenden Aussagenblöcke umfasst mehrere Einzelaussagen. Die sich unterscheidenden Aussagenbausteine sind durch Kästchen gekennzeichnet. Kreuzen Sie genau jene Bausteine an, die in einer wahren Einzelaussage enthalten sind. Jeder Aussagenblock enthält mindestens eine wahre Einzelaussage. Unvollständig oder falsch angekreuzte Aussagenblöcke werden mit null Punkten bewertet. Sie erhalten einen Punkt für jeden Aussagenblock, für den Sie genau die richtige Menge an Aussagenbausteinen angekreuzt haben.

Falls $\mathsf{P} \neq \mathsf{NP}$ gilt, dann

- ☒ gibt es keinen absoluten polynomiellen Approximationsalgorithmus für das Problem CLIQUE.
- ☒ gibt es keinen absoluten polynomiellen Approximationsalgorithmus für das Problem KNAPSACK.
- ☐ kann für einen Graphen $G = (V, E)$ und eine Knotenmenge $V' \subseteq V$ nicht in polynomieller Zeit überprüft werden, ob V' eine Clique in G ist.
- ☒ hält eine Turingmaschine, die eine semientscheidbare Sprache akzeptiert, nicht notwendigerweise auf jeder Eingabe.

Für jede Chomsky-2-Grammatik G in Chomsky-Normalform

- ☐ gibt es einen nicht-deterministischen endlichen Automaten, der $L(G)$ akzeptiert.
- ☒ kann der Cocke-Younger-Kasami Algorithmus für die Überprüfung, ob ein Wort in $L(G)$ liegt, verwendet werden.
- ☒ gibt es eine Turingmaschine, die $L(G)$ akzeptiert.
- ☐ enthält $L(G)$ unendlich viele Wörter.

Seien L_1 und L_2 zwei entscheidbare Sprachen über dem Alphabet Σ, dann ist auch die Sprache

- ☒ $L_1 \cup L_2$
- ☒ $L_1 \cap L_2$
- ☒ $L_1 \cdot L_2$
- ☒ $(\Sigma^\star \setminus L_1) \cup L_2$

entscheidbar.

Jede kontextfreie Grammatik G, die nicht das leere Wort erzeugt,

- ☐ besitzt für jedes ableitbare Wort einen eindeutigen Syntaxbaum.
- ☒ kann in eine Grammatik G' in Chomsky-Normalform überführt werden, sodass $L(G) = L(G')$.
- ☐ kann in einen endlichen Automaten $\mathcal{A}$ überführt werden, der die Sprache $L(G)$ akzeptiert.
- ☒ kann in eine Grammatik G' in Greibach-Normalform überführt werden, sodass $L(G) = L(G')$.

Jeder deterministische Kellerautomat $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, der durch einen Endzustand akzeptiert, hat die Eigenschaft,

- ☐ dass für alle $q \in Q$, $a \in \Sigma$ und $Z \in \Gamma$ gilt $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| > 1$.
- ☒ dass für alle $q \in Q$, $a \in \Sigma$ und $Z \in \Gamma$ gilt $|\delta(q, a, Z)| \leq 2$.
- ☐ dass $\mathcal{A}$ die Eingabe akzeptiert, wenn $\mathcal{A}$ in einen Zustand aus F wechselt.
- ☒ dass er nicht die Sprache $L = \{ww^R \mid w \in \{0, 1\}^\star\}$ akzeptiert, wobei w^R das Spiegelwort von w bezeichnet.

Für die Entropie einer diskreten, endlichen Zufallsvariable mit n Zeichen gilt:

- ☒ Sie wird maximal, wenn alle Zeichen gleichwahrscheinlich sind.
- ☒ Sie beträgt maximal $\log_2(n)$.
- ☐ Sie beträgt maximal $1/\log_2(n)$.
- ☐ Sie beträgt $n \cdot \log_2(n)$.