1. Klausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2024/2025

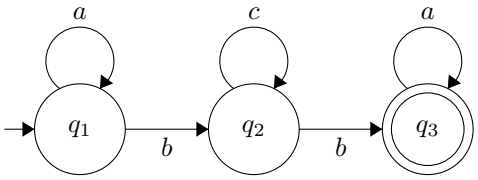
Lösung!

- Bringen Sie den Aufkleber mit Ihrer Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrer Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Am Ende der Klausur sind zusätzliche Leerseiten. Fordern Sie zusätzliches Papier bitte nur an, wenn nötig.
- Die Tackernadel darf nicht gelöst werden, Sie dürfen allerdings die NP-vollständigen Probleme im Anhang abtrennen.
- Begründen/Beweisen Sie Ihre Antworten ausreichend, wenn "Zeigen/Beweisen Sie, dass" gefordert wird.
- Als Hilfsmittel ist ein handbeschriebenes A4-Papier erlaubt.
- Einlesezeit: 15 min
  Bearbeitungszeit: 2 h

	Mögliche Punkte						Erreichte Punkte					
	a	b	c	d	e	Σ	a	b	c	d	e	Σ
Aufg. 1	1	2	2	4	–	9					–	
Aufg. 2	3	2	3	2	–	10					–	
Aufg. 3	1	4	2	–	–	7				–	–	
Aufg. 4	2	3	3	4	–	12					–	
Aufg. 5	2	2	7	–	–	11				–	–	
Aufg. 6	1	5	5	–	–	11				–	–	
Σ						60						

Problem 1: Warmup $1 + 2 + 2 + 4 = 9$ Punkte

Betrachten Sie den folgenden Automaten $\mathcal{A}$ mit implizitem Müllzustand über dem Alphabet $\{a, b, c\}$ und mit Startzustand q_1.



(a) Geben Sie für die Wörter $abccbab, abcccba$ jeweils an, ob sie zur Sprache $\mathcal{L}(\mathcal{A})$ gehören.

(b) Geben Sie einen DEA an, der $\mathcal{L}(\mathcal{A})^c$ akzeptiert.

(c) Geben Sie einen regulären Ausdruck α an, sodass $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\alpha)$.

(d) Wir definieren die Quotientensprache aus der Vorlesung wie gewohnt als

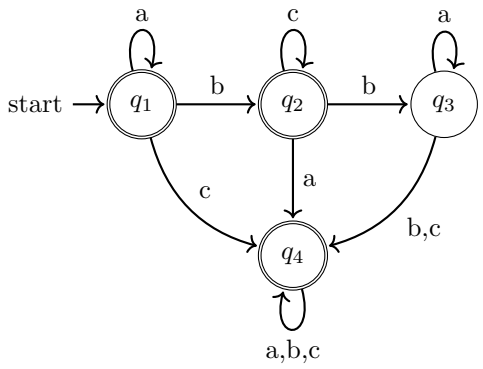
$$L_1/L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}.$$

Geben Sie einen regulären Ausdruck β für $\mathcal{L}(\mathcal{A})/\{ba\}$ an. Beweisen Sie, dass $\mathcal{L}(\beta) = \mathcal{L}(\mathcal{A})/\{ba\}$.

Lösung:

(a) *abccbab* ist nicht in der Sprache, *abcccbaa* ist in der Sprache.

(b)



(c) $\alpha = a^*bc^*ba^*$

(d) $L(a^*bc^*) = \mathcal{L}(\mathcal{A})/\{ba\}$.
'$\subseteq$': Für jedes Wort $w' \in L(a^*bc^*)$ gilt, dass $\delta^*(q_1, w') = q_2$, somit ist $\delta^*(q_1, w'ba) = q_3$ und somit $w' \in \mathcal{L}(\mathcal{A} \setminus \{ba\})$.
'$\supseteq$': Sei $w \in \mathcal{L}(\mathcal{A})/\{ba\}$, also $w \cdot ba \in \mathcal{L}(\mathcal{A})$. Der einzige akzeptierende Zustand im Automaten ist q_3, welcher ausschließlich von q_2 aus mit der Eingabe ba erreicht werden kann (von q_1, q_3 landen wir im Müllzustand). Somit gilt $\delta^*(q_1, w) = q_2$. Man sieht am Automaten, dass alle solche wörter von der Form a^*bc^* sind.

Problem 2: Kontextfreie Sprachen $3 + 2 + 3 + 2 = 10$ Punkte

Gegeben sei die Sprache $L := \{a^n b^m c^n b^\ell \mid n, m, \ell \geqslant 0\}$.

(a) Zeigen Sie, dass die Sprache L nicht regulär ist.

(b) Geben Sie eine kontextfreie Grammatik G an, die die Sprache L erzeugt.

(c) Geben Sie einen nichtdeterministischen Kellerautomaten $\mathcal{A}$ an, sodass $L = \mathcal{L}(\mathcal{A})$.

(d) Zeigen Sie, dass es eine Sprache $L' \subseteq L$ gibt, sodass L' nicht kontextfrei ist.

Hinweis: Benutzen Sie eine bereits bekannte nicht-kontextfreie Sprache aus der Vorlesung/Übung.

Lösung:

(a) Für beliebige $n, m, \ell \geqslant 1$ wählen wir $w = a^n b^m c^n b^\ell \in L$. Es gilt $|w| > n$. Für alle Zerlegung $uvx = a^n b^m c^n b^\ell$ mit $|uv| \leqslant n$, $v \neq \varepsilon$ gilt $u = a^\alpha, v = a^\beta$ für $\alpha + \beta \leqslant n$ und $\beta \geqslant 1$.

Es gilt

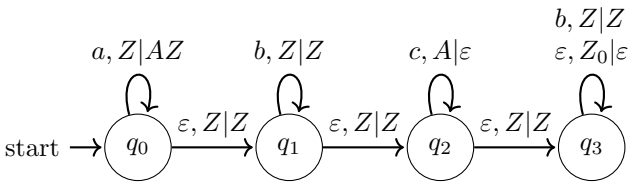
$$uv^0x = a^\alpha a^{n-\alpha-\beta} b^m a^n b^\ell = a^{n-\beta} b^m a^n b^\ell \notin L$$

da $n - \beta < n$, was ein Widerspruch zum Pumping Lemma darstellt. Damit ist die Sprache L nicht regulär.

(b) Wir geben die folgende Kontextfreie Grammatik für L: $G = (\Sigma = \{a, b, c\}, N = \{S, M, B\}, S, P)$ mit den Produktionsregel:

$$P = \left\{ \begin{array}{lcccl} S & \to & MB & | & \varepsilon, \\ M & \to & aMc & | & B, \\ B & \to & Bb & | & \varepsilon \end{array} \right\}$$

(c) Sei das Alphabet $\Sigma = \{a, b\}$ und das Arbeitsalphabet $\Gamma = \{Z_0, A\}$. Wir geben den Automaten $A = (\{q_0, q_1, q_2, q_3\}, \Sigma, \Gamma, \delta, q_0, Z_0)$ mit Akzeptanz durch leeren Stack. Sei $Z \in \Gamma$ ein beliebiges Stacksymbol:



(d) Wir setzen $m = 0$ und $\ell = n$ und nennen die resultierende Sprache L'. Offensichtlich gilt $L' \subseteq L$, da wir die Parameter in den erlaubten Intervallen gewählt haben. Es gilt also $L' = \{a^n b^n c^n \mid n \geqslant 1\} \subseteq L$, welche bekanntermaßen nicht kontextfrei ist.

Problem 3: Approximationsalgorithmen 1 + 4 + 2 = 7 Punkte

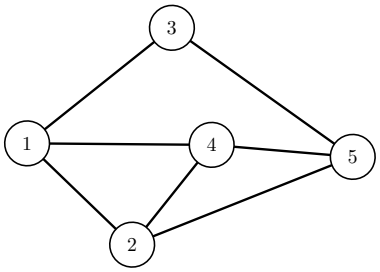
In dieser Aufgabe betrachten wir das Problem Vertex Cover: Gegeben einen Graphen, ist das Ziel, ein kleinstmögliches *Vertex Cover* zu finden, also eine Menge von Knoten, sodass jede Kante des Graphen mindestens einen der beiden Endpunkte in der Menge hat.

Vertex Cover
Gegeben: ein ungerichteter Graph $G = (V, E)$.
Aufgabe: Berechne die Größe eines minimalen Vertex Covers $C \subseteq V$, sodass für jede Kante $\{u, v\} \in E$ gilt: $u \in C$ oder $v \in C$.

Außerdem geben wir die folgenden Definition eines *maximalen Matching*:

Definition: Matching
Sei $G = (V, E)$ ein ungerichteter Graph. Ein *Matching* M ist eine Menge von Kanten, sodass keine zwei Kanten aus M einen gemeinsamen Endpunkt haben. Das heißt für alle e und e' aus M gilt: $e \cap e' = \emptyset$. Wir nennen ein Matching M *maximal* (nicht erweiterbar), wenn wir keine weitere Kante hinzufügen können: Es gilt für alle $e \in E - M$: $M \cup \{e\}$ ist kein Matching mehr.

(a) Geben Sie für den folgenden Graphen ein minimales Vertex Cover sowie ein maximales Matching an.



Gegeben sei der folgende Approximationsalgorithmus für Vertex Cover.

Algorithmus $\mathcal{A}$: Greedy Vertex Cover

Input: $G = (V, E)$
1 Berechne ein maximales Matching M in G
2 **for** $\{u, v\} \in M$ **do**
3 Füge u und v zu C hinzu
4 **return** $|C|$

(b) Zeigen Sie, dass der Algorithmus Greedy Vertex Cover eine Approximation mit relativer Gütegarantie 2 erreicht.

(c) Sei $n \in \mathbb{N}$ beliebig. Zeigen Sie, dass es einen Graphen mit n Knoten gibt, sodass ein minimales Vertex Cover Größe OPT hat, aber Greedy Vertex Cover als Ergebnis $2 \cdot OPT$ zurückliefert.

Lösung:

(a) Minimale Vertex Cover sind: $\{1, 4, 5\}$ oder $\{2, 3, 4\}$ oder $\{1, 2, 5\}$. Mögliche maximale Matchings sind $\{14, 25\}, \{14, 35\}, \{12, 45\}, \{12, 35\}, \{13, 24\}, \{13, 25\}, \{24, 35\}$.

(b) Sei $A(G)$ die Ausgabe des Algorithmus und sei M ein maximales Matching in G.

Wir zeigen dass $A(G)$ ein valides Vertex Cover ist. Da M ein *maximales* Matching ist, gilt für jede Kante $uv \in E$ dass entweder $u \in M$ oder $v \in M$, sonst kann M durch uv erweitert werden. Somit ist die Vertex Cover Bedingung erfüllt.

Wir zeigen nun, dass Greedy Vertex Cover eine Approximationsgüte von 2 erreicht. Zuerst zeigen wir, dass $OPT \geqslant |M|$ gilt. Angenommen $OPT < |M|$, dann gibt es eine Kante $\{u, v\} \in M$, für die u und v nicht im minimalen Vertex Cover enthalten sind, ein Widerspruch.

Damit gilt $|A(G)| = 2|M| \leqslant 2|OPT|$, da wir für jede Kante beide Endnoten zu C hinzufügen. Somit ist die relative Gütegarantie 2.

(c) Sei $K_{1,n}$ der Sterngraph mit n Knoten. Der Fall $n = 1$ enthält keine Kanten, daher betrachten wir $n > 1$: Es gilt $OPT = 1$ und $A(K_{1,n}) = 2$. Das minimale Vertex Cover beinhaltet den zentralen Knoten, der Algorithmus Greedy Vertex Cover findet ein maximales Matching mit genau einer Kante und gibt damit ein Vertex Cover der Größe 2 zurück.

Problem 4: Zeige oder Widerlege $2 + 3 + 3 + 4 = 12$ Punkte

Zeigen oder widerlegen Sie folgende Aussagen:

(a) Im Folgenden bezeichnen wir mit T_w die Turingmaschine mit Gödelnummer w.

Aussage: Die Sprache $\{w \in \{0,1\}^* : |w| < 2025 \text{ und } T_w \text{ akzeptiert } w \text{ nicht}\}$ ist entscheidbar.

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

(b) **Aussage:** Jeder DEA, der den regulären Ausdruck $a^* \cup b^*$ erkennt, hat mindestens 3 Zustände.

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

(c) Sei $\Sigma = \{a, b\}$. Für $k \in \mathbb{N}$ bezeichnen wir mit DEA^k die Klasse aller Sprachen $L \subseteq \Sigma^*$, für die es einen DEA ohne impliziten Müllzustand[1] gibt, der L akzeptiert und höchstens k Zustände hat.

Aussage: Für alle $k \in \mathbb{N}$ gilt: Die Klasse DEA^k ist unter Vereinigung abgeschlossen.

Tipp: Nutzen Sie Aussage (b).

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

(d) **Aussage:** Die Sprache $\{(w_1, w_2) : L(T_{w_1}) \subseteq L(T_{w_2})\}$ ist nicht semi-entscheidbar.

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

Möglicher Lösungsansatz: Benutzen Sie, dass die Sprache $\{(w_1, w_2) : L(T_{w_1}) = L(T_{w_2})\}$ aus der Übung nicht semi-entscheidbar ist.

Lösung:

(a) **Zeige:** Die Sprache ist endlich, da $|w| < 2025$ und somit regulär. Reguläre Sprachen sind aus der Vorlesung entscheidbar.

(b) **Zeige:** Wir betrachten die Nerode-Relation mit den folgenden (nicht notwendigerweise vollständigen, vorerst unbewiesenen) Äquivalenzklassen des Automaten: $[\varepsilon], [a], [b]$. Die Klassen $[\varepsilon]$,$[a]$ werden durch das Suffix b getrennt. Die Klassen $[\varepsilon]$,$[b]$ werden durch das Suffix a getrennt. Die Klassen $[a]$,$[b]$ werden durch das Suffix a getrennt. Damit muss der Minimalautomat mindestens 3 Zustände besitzen.

(c) **Widerlege:** Seien $L_a = a^*$ und $L_b = b^*$. Dann gibt es für beide Sprachen jeweils DEA mit 2 Zuständen. Für den DEA für L_a gibt es einen Automat mit Äquivalenklassen $[L_a] = a^*, [b] = \Sigma^* \setminus \{a^*\}$ die durch das Suffix a getrennt werden. Analog für L_b. Nach (b) gilt allerdings, dass der minimal DEA der Vereinigung mindestens 3 Zustände haben muss, demnach ist DEA^2 nicht unter Vereinigung abgeschlossen.

(d) **Zeige:** Wir nehmen an die Sprache sei semi-entscheidbar. Sei $L_= = \{(w_1, w_2) : L(T_{w_1}) = L(T_{w_2})\}$ die Äquivalenzsprache. Wir bauen eine Turingmaschine M' für $L_=$ wie folgt, wobei M die Maschine für die gegebene Sprache ist. Bei Eingabe (w_1, w_2) simuliere M auf Eingaben (w_1, w_2) sowie (w_2, w_1) und akzeptiere genau dann wenn M beide Wörter akzeptiert. Die Korrekheit folgt aus der Äquivalenz $A \subseteq B \wedge B \subseteq A \Leftrightarrow A = B$. M' ist semi-entscheidbar, da aus $A = B$ die Turingmaschine M für sowohl $A \subseteq B$ und $B \subseteq A$ nach Annahme der Semi-entscheidbarkeit dies korrekt verifizieren kann. Da $L_=$ wie aus der Übung bekannt nicht semi-entscheidbar ist, haben wir einen Widerrspruch erreicht und die gegebene Sprache kann nicht semi-entscheidbar sein.

[1] Die Übergangsfunktion $\delta : Q \times \Sigma \to Q$ muss total sein, d.h. für jedes Paar $(q, a) \in Q \times \Sigma$ ist $\delta(q, a)$ definiert.

Problem 5: NP-Vollständigkeit $2 + 2 + 7 = 11$ Punkte

Gegeben sei das folgende Problem:

KLEINER MENGENSCHNITT
Gegeben: Eine Grundmenge M, Teilmengen $L_1, \ldots, L_m \subseteq M$, sowie Zahlen $k, \ell \leqslant m$.
Problem: Gibt es eine Menge an Indizes $I \subseteq \{1, \ldots, m\}$ mit $|I| = k$, sodass

$$\left| \bigcap_{i \in I} L_i \right| \leqslant \ell?$$

(a) Gegeben sei folgende Instanz von KLEINER MENGENSCHNITT:

Grundmenge $M = \{1, 2, 3, 4, 5\}$ und Teilmengen

$$L_1 = \{1, 2, 3, 4\}, \quad L_2 = \{1, 2, 3\}, \quad L_3 = \{2, 4\}, \quad L_4 = \{2, 5\}, \quad L_5 = \{2, 3, 5\}$$

sowie $k = 3$ und $\ell = 1$. Zeigen Sie, dass es sich um eine Ja-Instanz handelt. Geben Sie, wenn möglich, einen Wert für ℓ an, sodass es sich um eine Nein-Instanz handeln würde.

(b) Zeigen Sie, dass KLEINER MENGENSCHNITT $\in$ NP.

(c) Zeigen Sie, dass KLEINER MENGENSCHNITT NP-schwer ist. Reduzieren Sie hierzu von einem geeigneten Problem im Anhang der Klausur auf KLEINER MENGENSCHNITT.

Hinweis: Es gilt $\left(\bigcup_{i \in I} L_i\right) = M - \bigcap_{i \in I} L_i^c$*, wobei* $-$ *die Mengendifferenz ist.*

Lösung:

(a) Wähle die folgenden 3 Mengen: L_2, L_3, L_4. Dann ist $|L_2 \cap L_3 \cap L_4| = |\{2\}| \leqslant 1$.

Für $\ell = 0$ ist die gegebene Instanz eine Nein-Instanz, da jede Menge das Element 2 enthält.

(b) Die OTM rät eine Menge $I \subseteq \{1, \ldots, m\}$ von Indizes der Teilmengen, sodass $|I| = k$ und $|\bigcap_{i \in I} L_i| \leqslant \ell$. Sollte eine dieser Eigenschaften nicht erfüllt sein, so lehnt die Turingmaschine ab, andernfalls akzeptiert sie.

Dieses Zertifikat kann in polynomieller Zeit verifiziert werden, da man für zwei Teilmengen $L, L' \subseteq M$ den Schnitt in $\mathcal{O}(|M|^2)$ berechnen kann und insgesamt sukzessive $k - 1$ Schnitte berechnen muss. $|I| = k$ sowie $I \subseteq \{1, \ldots, m\}$ lassen sich in Zeit $O(k)$ überprüfen. Schließlich überprüft die Turingmaschine in $\mathcal{O}(|M|)$, ob die Größe des resultierenden Schnittes kleiner oder gleich ℓ ist.

Da die Berechnung des Schnitts und die Überprüfung der Größe in polynomieller Zeit $\mathcal{O}(|M|^2 \cdot k)$ möglich ist, liegt KLEINER MENGENSCHNITT also in NP.

(c) Wir zeigen SET COVER $\propto$ KLEINER MENGENSCHNITT.
Transformation: Sei also eine Instanz von SET COVER gegeben, d.h. eine Grundmenge M und Teilmengen $U_1, \ldots, U_m \subseteq M$ sowie eine Zahl c. Wir konstruieren nun eine Instanz von KLEINER MENGENSCHNITT. Die Grundmenge bleibt weiterhin M. Wir setzen $k = c$, $\ell = 0$ und definieren Teilmengen $L_i = U_i^c$ für $i = 1, \ldots, m$ für die KLEINER MENGENSCHNITT Instanz.
Laufzeit: Die Transformation lässt sich in $\mathcal{O}(k \cdot |M|)$ durchführen, da wir für jedes U_i das Komplement berechnen müssen, das Kopieren der unveränderten Werte geschieht zusätzlich in Linearzeit, somit im gesamten in Polynomialzeit.
Korrektheit:

$\Longrightarrow$: Sei nun $I \subseteq \{1, \ldots, m\}$ eine Ja-Instanz von SET COVER mit $|I| \leqslant c$.

Mit dem Hinweis folgt, dass

$$\left(\bigcup_{i\in I} U_i\right) = M - \bigcap_{i\in I} U_i^c$$

Da I eine Ja-Instanz ist, gilt $\bigcup_{i\in I} U_i = M$ und somit $\bigcap_{i\in I} U_i^c = \emptyset$. Nach Konstruktion folgt also $\left|\bigcap_{i\in I} U_i^c\right| = \left|\bigcap_{i\in I'} L_i\right| \leqslant 0$ mit $I' = I$ und $|I'| \leqslant c$. Wir fügen nun noch $k - |I|$ viele, beliebige Indizes zu $|I'|$ hinzu. Da $\bigcap_{i\in I} L_i = \emptyset$ gilt folglich auch $\bigcap_{i\in I'} L_i = \emptyset$ denn $A \cap \emptyset = \emptyset$. Somit ist I' eine Ja-Instanz von Kleiner Mengenschnitt.

$\Longleftarrow$: Sei nun I' eine Ja-Instanz von Kleiner Mengenschnitt. Dann gilt $\left|\bigcap_{i\in I'} L_i\right| \leqslant 0$ und damit $\bigcap_{i\in I'} L_i = \emptyset$. Mit dem Hinweis gilt also, dass

$$M = M - \left(\bigcap_{i\in I'} L_i\right) = \bigcup_{i\in I'} L_i^c = \bigcup_{i\in I} U_i$$

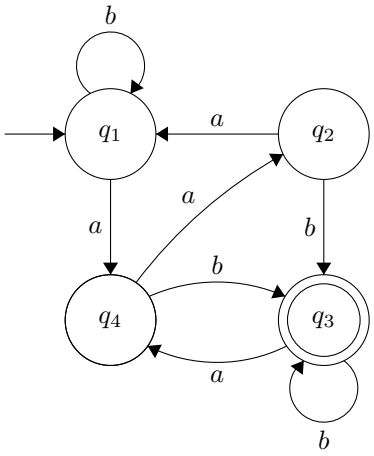
mit $I = I'$ und $c = k$. Damit ist I mit $|I| = k \leqslant c$ eine Ja-Instanz für Set Cover.

Problem 6: Synchronisierende DEAs $1 + 5 + 5 = 11$ Punkte

Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Wir nennen ein Wort $w \in \Sigma^*$ *synchronisierend* , falls man in jedem Zustand beim Einlesen des Wortes w in demselben Zustand q landet, d.h. es existiert ein Zustand $q \in Q$, sodass für alle Zustände $p \in Q$ gilt, dass $\delta^*(p, w) = q$.

Wir nennen einen DEA synchronisierend, falls für diesen DEA ein synchronisierendes Wort existiert. [2]

(a) Zeigen Sie, dass der folgende DEA synchronisierend ist, indem Sie ein synchronisierendes Wort w angeben.



(b) Beweisen Sie folgende Äquivalenz: Ein DEA A ist synchronisierend genau dann, wenn für jedes Zustandspaar $p_1, p_2 \in Q$ ein Wort $w \in \Sigma^*$ existiert, sodass

$$\delta^*(p_1, w) = \delta^*(p_2, w).$$

Achtung: Achten Sie auf die Quantorenreihenfolge!

(c) Geben Sie einen Polynomialzeitalgorithmus an, der für einen gegebenen DEA A entscheidet, ob dieser synchronisierend ist.

Hinweis: Sie dürfen das Resultat aus der Teilaufgabe b) benutzen. Es kann hilfreich sein, einen Produktautomaten zu konstruieren.

Lösung:

(a) ba (oder alle Wörter die ba als Teilwort haben)

(b) Wir zeigen beide Seiten der Äquivalenz.

Wenn A synchronisierend ist mit dem Wort w^*, ist w^* auch ein Wort um jedes Zustandspaar ineinander zu überführen.

Für die andere Richtung nehmen wir an, dass für alle Zustandspaare p_i, p_j ein synchronisierendes Wort w_{ij} existiert, und konstruieren ein global synchronisierendes Wort w^*. Für ein Wort w setzen wir die Menge Q_w als Menge der erreichbaren Zustände wenn alle Zustände Startzustände wären, formal $Q_w = \{\delta^*(q, w) \mid q \in Q\}$. Es gilt somit $Q_\varepsilon = Q$. Sei w unser aktuelles Wort und nehmen wir an dass $|Q_w| > 1$. Wir nehmen zwei Zustände $p_i, p_j \in Q_w$ und konkatenieren w mit w_{ij}. Da wir w_{ij} so wählen, dass zwei Zustände aus Q_w sich synchronisieren, muss $|Q_{ww_{ij}}| < |Q_w|$ gelten. Die

[2] Folgendes Problem ist ein offenes Problem in der Automatentheorie: Hat jeder synchronisierender DEA mit n Zuständen ein synchronisierendes Wort der Länge höchstens $(n-1)^2$?

Zustände aus Q_w upzudaten ist in Zeit $|Q_w| \cdot |w_{ij}|$ möglich und somit endlich. Wenn $|Q_{w^*}| = 1$, so ist w^* ein synchronisierendes Wort da nach Definition von Q_{w^*} alle Zustände nach Einlesen von w^* in einen Zustand überführt werden.

Wir können nun von $w = \varepsilon$ ausgehend wie oben beschrieben Q_w in jedem Schritt verringern bis $|Q_w| = 1$ erreicht ist. Da alle Schritte in endlicher Zeit laufen, muss dieser Algorithmus terminieren und wir erhalten ein synchronisierendes Wort für den gesamten Automaten.

(c) Wir betrachten den Algorithmus aus b. Wir beobachten, dass dieser in Zeit $O(|Q|^2 \cdot \max_{p_i,p_j \in Q}(|w_{ij}|))$ läuft, was polynomiell ist, sofern für alle $p_i, p_j \in Q$ das paarweise synchronisierende Wort w_{ij} polynomiell beschränkt ist.

Somit gilt nach b, dass wenn es für zwei beliebige Zustände p_i, p_j immer einen synchronisierendes Wort polynomieller Länge gibt, der Automat ein synchronisierendes Wort hat. Gibt es kein synchronisierendes Wort für ein Zustandspaar, so kann nach b der Automat auch nicht synchronisierend sein.

Wir beschreiben somit einen Algorithmus, der ein polynomiell beschränktes, synchronisierendes Wort für ein Zustandspaar p_i, p_j findet, sofern ein solches existiert. Wir betrachten den Produktautomaten von A mit sich selbst und setzen alle Zustände (p, p) für $p \in Q$ als akzeptierende Endzustände. Startzustand ist der Zustand (p_i, p_j). Ein akzeptierendes Wort ist nun synchronisierend für p_i, p_j:

Jedes akzeptierende Wort w_{ij} ist ein synchronisierendes Wort, da wir $\delta^*(p_i, w_{ij}) = \delta^*(p_j, w_{ij}) = p$ haben. Außerdem ist jedes synchronisierende Wort auch akzeptierend, da aus $\delta^*(p_i, w_{ij}) = \delta^*(p_j, w_{ij}) = p$ folgt $\delta^*((p_i, p_j), w_{ij}) = (p, p)$.

Konstruktion eines Produktautomaten ist in polynomieller Zeit möglich. Aus der Vorlesung ist bekannt, dass ein akzeptierendes Wort eines Automaten zu finden in Linearzeit möglich ist, welches somit linear (polynomiell) beschränkt ist.

Die folgenden Probleme können Sie als NP-vollständig annehmen und für Reduktionen verwenden.

SUBSET SUM
Gegeben: Zahlen $a_1 \dots a_n \in \mathbb{N}$, und Wert $t \in \mathbb{N}$.
Problem: Gibt es $I \subseteq \{1, \dots, n\}$, sodass $\sum_{i \in I} a_i = t$?

CLIQUE
Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k < |V|$.
Problem: Gibt es eine Clique der Größe mindestens k?

HAMILTON-KREIS
Gegeben: Ein ungerichteter Graph $G = (V, E)$.
Problem: Gibt es einen Kreis in G, der jeden Knoten genau einmal besucht?

3-COLOR
Gegeben: Ein ungerichteter Graph $G = (V, E)$.
Problem: Gibt es eine Funktion $c : V \to \{1, 2, 3\}$, sodass für jede Kante $\{u, v\}$ gilt $c(u) \neq c(v)$?

PARTITION
Gegeben: Zahlen $a_1 \dots a_n \in \mathbb{N}$.
Problem: Gibt es $I \subseteq \{1, \dots, n\}$, sodass $\sum_{i \in I} a_i = \sum_{j \notin I} a_j$?

3-SAT
Gegeben: Menge U an aussagenlogischen Variablen und eine Menge C von Klauseln mit genau drei Literalen.
Problem: Gibt es eine Wahrheitsbelegung der Variablen aus U, sodass alle Klauseln aus C erfüllt sind?

SAT
Gegeben: Menge U an aussagenlogischen Variablen und eine Menge C von Klauseln.
Problem: Gibt es eine Wahrheitsbelegung der Variablen aus U, sodass alle Klauseln aus C erfüllt sind?

SET COVER
Gegeben: Eine Grundmenge M, Teilmengen $U_1, \dots, U_m \subseteq M$ mit $\bigcup_{i=1}^{m} U_i = M$, sowie eine Zahl c.
Problem: Gibt es eine Indexmenge $I \subseteq \{1, \dots, m\}$ mit $|I| \leqslant c$ und $\bigcup_{i \in I} U_i = M$?

DOMINATING SET
Gegeben: Ein Graph $G = (V, E)$ und eine Zahl k.
Gesucht: Eine Teilmenge $D \subseteq V$, sodass $|D| \leqslant k$ und für alle Knoten $v \in V$ gilt $v \in D$ oder v hat einen Nachbarn $u \in D$, also $\{u, v\} \in E$.