Grundzüge der Theoretischen Informatik, WS 21/22: Musterlösung zur Probeklausur

Julian Dörfler

Lesen Sie bitte zuerst folgende Hinweise!

1. Benutzen Sie bitte einen blauen oder schwarzen nicht-löschbaren Stift.
2. Schreiben Sie bitte auf jedes Blatt Ihren Namen und Ihre Matrikelnummer.
3. Fangen Sie bitte jede Aufgabe auf einem neuen Blatt an.
4. Geben Sie bitte pro Aufgabe nur einen Lösungsversuch ab. Streichen Sie nicht gültige Lösungsversuche deutlich durch.
5. Sie dürfen auf alle Ergebnisse der Vorlesung in den Kapiteln 1 bis 30 und auf die Aufgaben der Übungsblätter 1 bis 13 und Präsenzblätter 1 bis 14 Bezug nehmen, außer dies wird in der Aufgabenstellung ausgeschlossen.
6. Ihr Merkblatt ist nicht Teil Ihrer Lösung. Verweise auf Ihr Merkblatt werden daher nicht gewertet.

Name: _____

Matrikelnummer: _____

Aufgabe	max.	erreicht
1	15	
2	16	
3	15	
4	13	
5	8	
Σ	67	

Aufgabe 1. *(15 Punkte)*
Welche der folgenden Aussagen sind richtig, welche falsch. Beweisen Sie Ihre Antworten.

(a) *(3 Punkte)* Es existiert eine Bijektion von V_0 auf H_0.

(b) *(3 Punkte)* Seien $A, B \subseteq \Sigma^\star$, wobei $A \notin \mathsf{REG}$ und B endlich ist. Dann ist ebenfalls $A \cup B \notin \mathsf{REG}$.

(c) *(3 Punkte)* Sei $L \leq_{\mathrm{P}} L'$ und $L' \in \mathsf{REG}$. Dann ist $L \in \mathsf{REG}$.

(d) *(3 Punkte)* Mindestens eine der Inklusionen $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$ ist strikt.

(e) *(3 Punkte)* Sei $L \subseteq H_0$ unendlich. Dann ist L unentscheidbar.

Solution 1.

(a) Richtig, solch eine Bijektion existiert, da sowohl V_0, als auch H_0 abzählbar unendlich sind und somit eine Bijektion $V_0 \to \mathbb{N}$ und eine Bijektion $\mathbb{N} \to H_0$ existiert. Die Komposition dieser beiden Bijektionen ist eine Bijektion $V_0 \to H_0$.

(b) Richtig, angenommen $A \cup B$ wäre regulär. Dann wäre $A = (A \cup B) \setminus (B \setminus A)$ aber ebenfalls regulär durch die Abschlusseigenschaften von REG und da $B \setminus A$ als endliche Menge regulär ist. Dies ist aber ein Widerspruch zur Annahme, dass $A \notin \mathsf{REG}$.

(c) Falsch. Sei $L' = \{\varepsilon\} \in \mathsf{REG}$ und $L = \{0^n 1^n \mid n \in \mathbb{N}\} \notin \mathsf{REG}$. Dann gilt aber trotzdem $L \leq L'$, denn bei Eingabe x kann die Reduktion einfach in Polynomialzeit entscheiden, ob $x \in L$ ist und dann auf $\varepsilon \in L'$ abbilden und ansonsten auf $0 \notin L'$. .

(d) Richtig. Angenommen keine dieser Inklusionen ist strikt, dann gilt $\mathsf{L} = \mathsf{PSPACE}$. Dies ist aber ein Widerspruch zum Platzhierarchiesatz, da $\log n \in o(p(n))$ für jedes nicht konstante Polynom p.

(e) Falsch. Wir betrachten das Subset L der `WHILE`-Programme der Form $x_0 := c$ für alle konstanten $c \in \mathbb{N}$. Dann halten diese auf jeder Eingabe, also insbesondere ihrer eigenen Gödelnummer. Somit gilt $L \subseteq H_0$ und L ist unendlich. L ist jedoch trivial entscheidbar.

Aufgabe 2. (Reguläre Sprachen) *(16 Punkte)*

(a) *(4 Punkte)* Geben Sie einen minimalen totalen DEA für folgende Sprache an und beweisen Sie dessen Minimalität, indem Sie Repräsentanten aller Myhill-Nerode-Äquivalenzklassen angeben und paarweise beweisen, dass diese nicht Myhill-Nerode-Äquivalent sind:

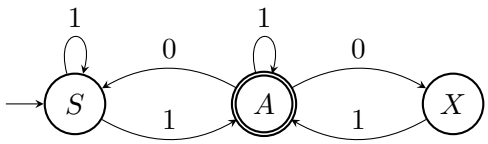
$$L_1 = \{\text{bin}(n) \in \{0,1\}^\star \mid n \equiv 0 \mod 2\}$$

Hierbei enthält $\text{bin}(n)$ keine führenden Nullen und es gilt $\text{bin}(0) = 0$.

Hinweis: Ihr totaler DEA sollte 5 Zustände haben.

(b) *(4 Punkte)* Konstruieren Sie zu folgendem Automaten einen äquivalenten deterministischen endlichen Automaten. Führen Sie dazu die Potenzmengenkonstruktion explizit durch. Vereinfachen[1] Sie *danach* den Automaten, wenn möglich, und geben Sie einen regulären Ausdruck für die akzeptierte Sprache an.

Hinweis: Sie können in der expliziten Konstruktion auf den Zustand, der der leeren Menge entspricht, verzichten. Nehmen Sie aber *keine* weiteren Vereinfachungen des Potenzmengen-Automaten vor. Geben Sie den neuen Zuständen sinnvolle Namen, nicht etwa A, B, C, ...



(c) *(4 Punkte)* Wir betrachten die Sprache

$$L_2 = \{0^n 1 2^m \mid n \neq m\}\,.$$

Beweisen Sie, dass $L_2 \notin \mathsf{REG}$.

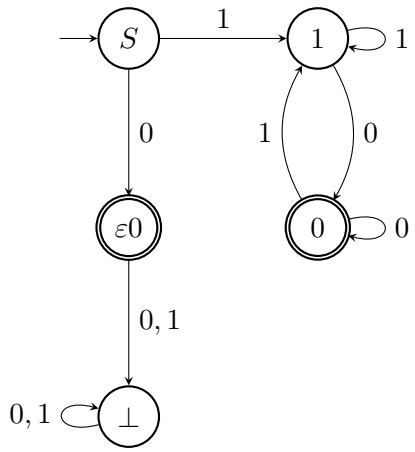
(d) *(3 Punkte)* Beweisen Sie, dass L_2 aus der vorherigen Teilaufgabe jedoch kontextfrei ist, indem Sie eine kontextfreie Grammatik angeben, die L_2 erkennt. Erklären Sie Ihre Konstruktion kurz.

(e) *(1 Punkt)* Können Sie auch eine linkslineare Grammatik für L_2 angeben? Begründen Sie Ihre Antwort.

[1] D.h. nicht erreichbare Zustände sollen entfernt werden.

Solution 2. (Reguläre Sprachen)

(a) Der folgende totale DEA erkennt L_1:



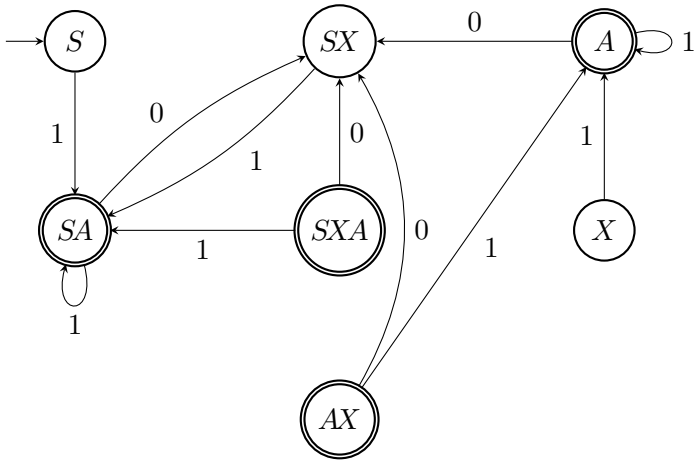
Um die Minimalität dieses DEAs zu beweisen geben wir 5 Respräsentanten unterschiedlicher Myhill-Nerode-Äquivalenzklassen an:

$$R = \{\varepsilon, 0, 1, 10, 00\}$$

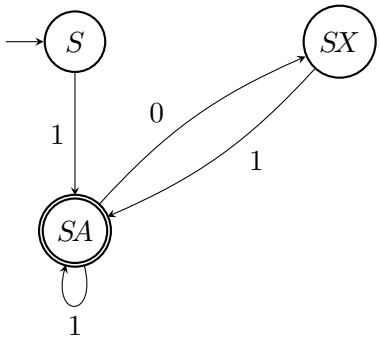
Für $x \neq y \in R$ beweisen wir $x \not\sim_{L_1} y$ indem wir eine Fortsetzung $b \in \{0,1\}^\star$ angeben, sodass genau ein Wort aus $\{xb, yb\}$ in L_1 enthalten ist. Diese fassen wir in folgender Tabelle zusammen:

	ε	0	1	10	00
ε	-	ε	00	ε	0
0		-	ε	0	ε
1			-	ε	0
10				-	ε
00					-

(b) Der Potenzmengenautomat hat die folgenden Zustände:
$\emptyset, \{S\}, \{A\}, \{X\}, \{SA\}, \{SX\}, \{AX\}, \{SAX\}$. Auf $\emptyset$ wurde der Einfachheit halber verzichtet.



Nach Entfernen nicht erreichbarer Zustände erhalten wir folgenden Automaten.



Für die akzeptierte Sprache L gilt $L = L(1(1+01)^*)$.

(c) Wir verwenden den Homomorphismus h definiert durch

$$h(0) = 0$$
$$h(1) = \varepsilon$$
$$h(2) = 1$$

Nun erhalten wir $h(L_2) = \{0^n 1^m \mid n \neq m\}$, welche nach Aufgabe P4.1(f) nicht regulär ist. Da sich Regularität aber unter Homomorphismen erhält, kann also L_2 ebenfalls nicht regulär sein.

Alternative Lösung: Wir zeigen dies indem wir unendlich viele Myhill-Nerode-Äquivalenzklassen nachweisen. Hierzu betrachten wir 0^i1 und 0^j1 für $i \neq j$. Dann ist $0^i12^i \notin L_2$, aber $0^j12^i \in L_2$. Somit ist $0^i1 \not\sim_{L_2} 0^j1$ und L_2 hat unendlich viele Myhill-Nerode-Äquivalenzklassen, da $i, j \in \mathbb{N}$ beliebig waren. Somit ist L_2 nicht regulär.

(d) Wir behandeln die Fälle $n > m$ und $n < m$ unabhängig in den Variablen G und K.

Die durch die folgenden Produktionen induzierte Grammatik erzeugt L_2:

$$\begin{aligned} S &\to 0G \mid K2 \\ G &\to 1 \mid 0G2 \mid 0G \\ K &\to 1 \mid 0K2 \mid K2 \end{aligned}$$

(e) Nein, dies ist unmöglich, da nach Aufgabe A13.5 links-und rechtslineare Grammatiken die selben Sprachen beschreiben. Wir hatten in der Vorlesung ebenfalls behandelt, dass rechtslineare Grammatiken genau die regulären Sprachen beschreiben. Somit ist die Existenz einer linkslinearen Grammatik für L_2 ein Widerspruch zu $L_2 \notin \mathsf{REG}$.

Aufgabe 3. (Berechenbarkeitstheorie) *(15 Punkte)*
Zur Erinnerung: Eine Sprache $U \subseteq \mathbb{N}$ ist letztendlich periodisch, wenn es $n_0, p \in \mathbb{N}$ mit $p > 0$ gibt, so dass für alle $n \geq n_0$ gilt: $n \in U \Leftrightarrow n + p \in U$.

Betrachten Sie die folgende Sprache

$$B = \{i \in \mathbb{N} \mid \operatorname{im} \varphi_i \text{ ist letztendlich periodisch}\}.$$

(a) *(2 Punkte)* Zeigen Sie: B ist eine nicht-triviale Indexmenge.

(b) *(1 Punkt)* Zeigen oder widerlegen Sie: $B \in \mathsf{REC}$.

(c) *(4 Punkte)* Zeigen oder widerlegen Sie: $B \in \mathsf{RE}$.

(d) *(4 Punkte)* Zeigen oder widerlegen Sie: $B \in \mathsf{co\text{-}RE}$.

(e) *(4 Punkte)* Sei $g \in B$. Zeigen Sie: Es gibt ein $i \in \mathbb{N}$, so dass

$$\forall x \in \mathbb{N} : \varphi_i(x) = \begin{cases} 1 & i \cdot x \in \operatorname{im} \varphi_g \\ 0 & \text{sonst} \end{cases}.$$

Solution 3. (Berechenbarkeitstheorie)

(a) Sei $i \in B$ und $j \in \mathbb{N}$ mit $\varphi_i = \varphi_j$. Dann gilt $\operatorname{im} \varphi_j = \operatorname{im} \varphi_i$, also ist $\operatorname{im} \varphi_j$ auch letztendlich periodisch, also $j \in B$. Ferner gilt, dass die Gödelnummer eines Programmes P welches die Quadratfunktion berechnet nicht in B enthalten ist, da die Menge aller Quadratzahlen nicht letztendlich periodisch ist (wir hatten auf den Übungsblättern schon gezeigt, dass $\{0^{n^2} \mid n \in \mathbb{N}\}$ nicht regulär ist, was äquivalent ist) und $\Omega \in B$ (das Bild der Überall undefinierten Funktion ist leer, also letztendlich periodisch).

(b) Aus (a) und dem Satz von Rice folgt $B \notin \mathsf{REC}$.

(c) Es gilt $B \notin \mathsf{RE}$. Wir reduzieren H_0 auf $\overline{B}$:

$$f(i) = \text{göd}(P_i)$$

wobei P_i wie folgt definiert ist:

Gegeben m, führe i auf i aus. Danach: Gib m^2 aus.

Sei $i \in H_0$. Dann hält i auf Eingabe i und $\operatorname{im} \varphi(P_i) = \operatorname{im} \varphi(P) = \{n^2 \mid n \in \mathbb{N}\}$, welche wie oben erwähnt nicht letztendlich periodisch ist, also $f(i) \notin B$ und damit $f(i) \in \overline{B}$.
Sei $i \notin H_0$. Dann hält i nicht auf i und $f(i) = \text{göd}(P_i)$. Dann gilt aber, dass $\varphi_{\text{göd}(P_i)} = \varphi_\Omega = \emptyset$, welche letztendlich periodisch ist, also $f(i) \in B$ und damit $f(i) \notin \overline{B}$.

Da $H_0 \notin \mathsf{co\text{-}RE}$ und f offensichtlich `WHILE`-berechenbar ist, ist $\overline{B} \notin \mathsf{co\text{-}RE}$, also $B \notin \mathsf{RE}$.

(d) Es gilt $B \notin \mathsf{co\text{-}RE}$. Wir reduzieren H_0 auf B:

$$f(i) = \text{göd}(P_i)$$

wobei P_i wie folgt definiert ist:
Gegeben m, führe i auf i für m Schritte aus. Wenn die Simulation bis dahin nicht terminiert, gib m^2 aus, sonst divergiere.
Sei nun $i \in H_0$. Dann existiert c, so dass i nach genau c Schritten auf i hält. Dann ist $\operatorname{im} \varphi_{\text{göd}(P_i)}$ endlich, also letztendlich periodisch (z.B. wenn wir $n_0 \geq c$ wählen.) Folglich ist $f(i) = \text{göd}(P_i) \in B$.
Sei $i \notin H_0$. Dann ist $f(i) = \text{göd}(P_i)$ und $\varphi_{\text{göd}(P_i)} = \varphi_{\text{göd}(P)}$ ist wieder nicht letztendlich periodisch. Also ist $f(i) \notin B$.

Da $H_0 \notin \mathsf{co\text{-}RE}$ und f offensichtlich `WHILE`-berechenbar ist, ist $B \notin \mathsf{co\text{-}RE}$.

(e) Wir betrachten die Funktion

$$f(i,x) = \begin{cases} 1 & i \cdot x \in \operatorname{im} \varphi_g \\ 0 & \text{sonst} \end{cases}.$$

Zuerst zeigen wir, dass f `WHILE`-berechenbar ist. Hierzu reicht es zu zeigen, dass `WHILE`-Programme $y \in \operatorname{im} \varphi_g$ entscheiden können. Wir bemerken zuerst, dass $\operatorname{im} \varphi_g$ letztendlich periodisch ist. Das folgende `WHILE`-Programm entscheidet nun, ob $y \in \operatorname{im} \varphi_g$ ist:

Bei Eingabe y prüfe zuerst ob $y < n_0$. Falls dies der Fall ist, gib eine hardgecodete Antwort aus, ob $y \in \operatorname{im} \varphi_g$. Andernfalls berechne das minimale $y' \in \mathbb{N}$, so dass $y' \geq n_0$ und $y = y' + k \cdot p$ für irgendein $k \in \mathbb{N}$. Nun gib die hardgecodete Antwort aus, ob $y' \in \operatorname{im} \varphi_g$.

Wir müssen hierzu nur $n_0 + p$, also endlich viele, Antworten hardcoden, somit ist dies ein gültiges `WHILE`-Programm.

Die Aussage folgt nun mit dem Rekursionstheorem.

Aufgabe 4. *(13 Punkte)*
Eine aussagenlogische Formel ϕ in konjunktiver Normalform (CNF) heißt *positiv*, wenn ϕ keine Negationen enthält. Die Formel

$$\phi_1 = (x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4)$$

ist beispielsweise eine positive CNF. Die Formel

$$\phi_2 = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)$$

ist sogar eine positive 2CNF. Wir betrachten die folgenden beiden Probleme:

$$\mathsf{PosSAT} := \{\phi \mid \phi \in \mathsf{SAT} \wedge \phi \text{ ist eine positive CNF}\}$$

$$\mathsf{WPos2SAT} := \{(\phi, k) \mid \phi \text{ ist eine positive 2CNF und hat eine erfüllende Belegung mit } \textbf{genau } k \text{ Einsen}\}$$

Zur Veranschaulichung betrachten wir ϕ_1 und ϕ_2. Da beide erfüllbare positive CNFs sind, gilt $\phi_1 \in \mathsf{PosSAT}$ und $\phi_2 \in \mathsf{PosSAT}$. Ferner gilt, dass (ϕ_1, k) für **kein** k in $\mathsf{WPos2SAT}$ enthalten ist, da ϕ_1 keine 2CNF ist. ϕ_2 dagegen ist eine 2CNF und kann mit der Belegung $x_1 = x_2 = 1$ und $x_3 = 0$ erfüllt werden. Da diese Belegung genau zwei Einsen hat, gilt $(\phi_2, 2) \in \mathsf{WPos2SAT}$. Es gibt jedoch keine erfüllende Belegung von ϕ_2 mit nur einer Eins, daher gilt $(\phi_2, 1) \notin \mathsf{WPos2SAT}$.

(a) *(1 Punkt)* Zeigen Sie, dass $\mathsf{PosSAT} \in \mathsf{P}$.

(b) *(7 Punkte)* Zeigen Sie, dass $\mathsf{WPos2SAT}$ NP-vollständig ist.
Tipp: Reduzieren Sie von VC.

(c) *(5 Punkte)* Eine CNF heißt *negativ*, wenn jedes Literal eine negierte Variable ist. Definieren Sie NegSAT und $\mathsf{WNeg2SAT}$ analog und zeigen Sie, dass

- $\mathsf{NegSAT} \in \mathsf{P}$ und, dass
- $\mathsf{WNeg2SAT}$ NP-vollständig ist.

Hinweis: Nutzen Sie dafür Aufgabenteil (b), auch wenn Sie diesen nicht bearbeitet oder gelöst haben oder reduzieren Sie von IS.

Solution 4.

a) Trivial: Jede positive Formel in konjunktiver Normalform ist erfüllt, wenn alle Variablen auf 1 gesetzt werden.

b) Wir zeigen erst, dass $\mathsf{WPos2SAT} \in \mathsf{NP}$ indem wir eine nichtdeterministische Turingmaschine konstruieren: Gegeben eine Formel ϕ in positiver 2CNF und ein $k \in \mathbb{N}$, raten wir k Variablen von ϕ. Dann setzten wir diese Variablen auf 1 und alle anderen auf 0 und prüfen ob die Belegung erfüllend ist. Wenn ja akzeptieren wir, sonst verwerfen wir.

Nun reduzieren wir von VC: Gegeben eine VC-Instanz (G, k), konstruieren wir ϕ_G wie folgt: Jeder Knoten v von G wird eine Variable x_v. Dann fügen wir für jede Kante $\{u, v\}$ die Klausel $(x_u \vee x_v)$ ein. Schließlich gibt unsere Reduktion (ϕ_G, k) zurück. Diese Konstruktion kann offensichtlich in polynomieller Zeit durchgeführt werden.
Sei nun $(G, k) \in \mathsf{VC}$. Dann gibt es einen Vertex Cover C der Größe k. Dann erfüllt die Belegung, welche jede Variable x_v mit $v \in C$ auf 1 setzt und alle anderen auf 0 die Formel ϕ_G.
Sei $(\phi_G, k) \in \mathsf{WPos2SAT}$. Dann gibt es eine erfüllende Belegung a von ϕ_G, die genau k Variablen auf 1 setzt. Dann ist die Menge $C = \{v \mid a(x_v) = 1\}$ ein k-Vertex Cover von G.

c) Wir definieren

$$\mathsf{NegSAT} := \{\phi \mid \phi \in \mathsf{SAT} \wedge \phi \text{ ist eine negative CNF}\}$$

$$\mathsf{WNeg2SAT} := \{(\phi, k) \mid \phi \text{ ist eine negative 2CNF und hat eine erfüllende Belegung mit } \textbf{genau } k \text{ Einsen}\}$$

Hinweis: Die folgende Lösung ist genauso ausführlich wie in Aufgabenteil 4 (b). In der Klausur hätte man sich kurz fassen können, da die Reduktion analog zur Reduktion von Vertex-Cover funktioniert.

- $\mathsf{NegSAT} \in \mathsf{P}$ ist ebenfalls trivial: Setze alle Variablen auf 0, dann ist die Formel erfüllt.
- Wir zeigen erst, dass $\mathsf{WNeg2SAT} \in \mathsf{NP}$ indem wir eine nichtdeterministische Turingmaschine konstruieren: Gegeben eine Formel ϕ in negativer 2CNF und ein $k \in \mathbb{N}$, raten wir k Variablen von ϕ. Dann setzten wir diese Variablen auf 1 und alle anderen auf 0 und prüfen ob die Belegung erfüllend ist. Wenn ja akzeptieren wir, sonst verwerfen wir.
  Nun reduzieren wir von IS: Gegeben eine IS-Instanz (G, k), konstruieren wir ϕ_G wie folgt: Jeder Knoten v von G wird eine Variable x_v. Dann fügen wir für jede Kante $\{u, v\}$ die Klausel $(\neg x_u \vee \neg x_v)$ ein. Schließlich gibt unsere Reduktion (ϕ_G, k) zurück. Diese Konstruktion kann offensichtlich in polynomieller Zeit durchgeführt werden.
  Sei nun $(G, k) \in \mathsf{IS}$. Dann gibt es ein Independent Set I der Größe k. Dann erfüllt die Belegung, welche jede Variable x_v mit $v \in I$ auf 1 setzt und alle anderen auf 0 die Formel ϕ_G.
  Sei $(\phi_G, k) \in \mathsf{WNeg2SAT}$. Dann gibt es eine erfüllende Belegung a von ϕ_G, die genau k Variablen auf 1 setzt. Dann ist die Menge $I = \{v \mid a(x_v) = 1\}$ ein k-Vertex Cover von G.

Aufgabe 5. (Die Independent-Set-Vermutung) *(8 Punkte)*
Erinnern Sie sich, dass die Funktion

$$f : (G, k) \mapsto (G, |V(G)| - k)$$

eine Polynomialzeit-many-one-Reduktion von IS nach VC ist. Vielleicht haben Sie sich gefragt, ob es auch eine andere Reduktion von IS nach VC gibt, die G verändert, aber nicht k. In dieser Aufgabe sollen Sie zeigen, dass dies *wahrscheinlich* nicht möglich ist. Betrachten Sie dazu die folgende komplexitätstheoretische Vermutung[2]

> ISC*: Es gibt keine berechenbare Funktion g, so dass es einen (deterministischen) Algorithmus A gibt, der gegeben einen Graphen G und eine natürliche Zahl k korrekt entscheidet ob $(G, k) \in$ IS und dessen Laufzeit beschränkt ist durch*
>
> $$g(k) \cdot |V(G)|^{O(1)}.$$

(a) *(2 Punkte)* Zeigen Sie: ISC $\Rightarrow$ P $\neq$ NP.

(b) *(3 Punkte)* Zeigen Sie: Es gibt einen Algorithmus A' der gegeben (G, k) korrekt entscheidet ob $(G, k) \in$ VC und dessen Laufzeit beschränkt ist durch

$$O(k)^{O(k)} \cdot |V(G)|^{O(1)}.$$

Tipp: Stellen Sie sich vor, Sie entfernen schrittweise Kanten (mitsamt Knoten) aus G. Angenommen Sie haben auf diese Weise k Kanten entfernt und es gibt immer noch verbleibende Kanten in G, kann G dann noch einen Vertex-Cover der Größe k haben?

(c) *(3 Punkte)* Nutzen Sie (b) um zu zeigen, dass unter der Annahme ISC folgendes gilt:

> *Für KEINE Polynomialzeit-many-one-Reduktion*
>
> $$f : \Sigma^* \to \Sigma^*$$
> $$(G, k) \mapsto (G', k')$$
>
> *von* IS *nach* VC *gibt es eine berechenbare Funktion g, so dass für alle $(G, k) \in \Sigma^*$ mit $f(G, k) = (G', k')$ gilt, dass $k' \leq g(k)$.*

In anderen Worten: k' hängt immer auch von $|V(G)|$ ab.

[2] Diese Vermutung (ISC) wird von der sogenannte *Exponential Time Hypothesis (ETH)* impliziert. ETH ist eine stärkere Vermutung als P $\neq$ NP — in dem Sinne, dass ETH $\Rightarrow$ P $\neq$ NP, die Rückrichtung aber nicht bekannt ist. Trotzdem wird ETH, genau wie P $\neq$ NP von der Community als wahr angenommen, da ein Beweis, dass ETH nicht gilt fast ebenso weitreichende Folgen wie P = NP hätte.

Solution 5. (Die Independent-Set-Vermutung) (a) Wir zeigen die Kontraposition. Sei also $\mathsf{P} = \mathsf{NP}$, dann gibt es eine Konstante c und einen Algorithmus A, der IS in Zeit $(|k| + |V(G)|)^c$ entscheidet. Nun gilt aber

$$(|k| + |V(G)|)^c \leq (2 \cdot |k| \cdot |V(G)|)^c = (2 \cdot |k|)^c \cdot |V(G)|^c .$$

Die Aussage folgt, da $k \mapsto (2 \cdot |k|)^c$ offensichtlich berechenbar ist.

(b) Unser Algorithmus entfernt erst isolierte Knoten. Danach führen wir die Instruktion aus dem Hinweis k mal aus. Angenommen G hat einen k-Vertex-Cover. Dann muss dieser Vertex-Cover von jeder der k (paarweise nicht inzidenten) Kanten mindestens einen Knoten beinhalten. Hat also G nach k Schritten immer noch mindestens eine Kante, dann kann G keinen k-Vertex-Cover haben und unser Algorithmus verwirft. Ansonsten seien $e_1, \ldots, e_k$ die entfernten Kanten und V_k die Knotenmenge dieser Kanten. Es gilt, dass $|V_k| = 2k$. Wir testen nun für jede Teilmenge der Größe k von V_k ob diese ein Vertex-Cover von G ist. Die Gesamtlaufzeit ist dann offensichtlich beschränkt durch

$$O(k)^{O(k)} \cdot |V(G)|^{O(1)} .$$

(c) Wir nehmen an, dass eine solche Reduktionsfunktion existiert und konstruieren einen Algorithmus A für IS:
Gegeben (G, k), berechnen wir zunächst $f(G, k) = (G', k')$ und führen dann den Algorithmus A' aus (b) aus.
Die Korrektheit folgt aus (b) und der Annahme, dass f eine Polynomialzeit-many-one-Reduktion ist. Interessanter ist die Laufzeitanalyse: Es existiert eine Konstante d, so dass f in Zeit $(|k| + |V(G)|)^d$ berechenbar ist. Es folgt, dass $|V(G')| \leq (|k| + |V(G)|)^d$ (sonst bräuchte man schon mehr Zeit um G' überhaupt aufzuschreiben). Ferner wissen wir aus der Annahme, dass es eine berechenbare Funktion g gibt, so dass $k' \leq g(k)$. Nun nutzen wir die Laufzeitschranke von A' aus (b) — es existieren also Konstanten a, b und c, so dass VC in Zeit $(ak')^{(bk')} \cdot |V(G')|^c$ gelöst werden kann — und erhalten eine Gesamtlaufzeit von

$$\begin{aligned}
&(|k| + |V(G)|)^d + (a \cdot g(k))^{(b \cdot g(k))} \cdot ((|k| + |V(G)|)^d)^c \\
\leq &(|k| + |V(G)|)^d + (a \cdot g(k))^{b \cdot g(k)} \cdot (2|k||V(G)|)^{dc} \\
\leq &(2|k||V(G)|)^d + (a \cdot g(k))^{b \cdot g(k)} \cdot (2|k|)^{dc} \cdot (|V(G)|)^{dc} \\
\leq &(2|k|)^d \cdot |V(G)|^d + (a \cdot g(k))^{b \cdot g(k)} \cdot (2|k|)^{dc} \cdot |V(G)|^{dc} \\
\leq &2 \cdot \left((2|k|)^d \cdot |V(G)|^d \cdot (a \cdot g(k))^{b \cdot g(k)} \cdot (2|k|)^{dc} \cdot |V(G)|^{dc} \right) \\
= &\left(2 \cdot (2|k|)^{d+dc} \cdot (a \cdot g(k))^{b \cdot g(k)} \right) \cdot \left(|V(G)|^{d+dc} \right) \in O\left(g'(k) \cdot |V(G)|^{O(1)} \right) ,
\end{aligned}$$

wobei

$$g'(k) := 2 \cdot (2|k|)^{d+dc} \cdot (a \cdot g(k))^{b \cdot g(k)}$$

berechenbar ist, da auch g per Annahme berechenbar ist. Folglich ist ISC falsch, was ein Widerspruch zur Annahme der Existenz von f ist. Folglich gilt die Aussage.