Übungen zur Vorlesung Grundbegriffe der Theoretischen Informatik

Thomas Schwentick

Gaetano Geck, Lutz Oettershagen,
Christopher Spinrath, Marco Wilhelm

technische universität dortmund

Lehrstuhl 1
Logik in der Informatik

SoSe 2018 | **Übungsblatt 12** | **03.07.2018**

Aufgabe 12.1 [NP] **6 Punkte**

Wir untersuchen die Komplexität der Probleme GraphIsomorphismus und FinishSat, die wie folgt definiert sind.

Problem: GraphIsomorphismus
Gegeben: Gerichtete Graphen G und G'
Frage: Sind die Graphen isomorph?

Problem: FinishSat
Gegeben: Eine aussagenlogische Formel φ in KNF und eine partielle Wahrheitsbelegung $\alpha : \{x_1, x_2, \ldots\} \rightharpoonup \{0, 1\}$
Frage: Lässt sich α zu einer totalen Funktion $\beta : \{x_1, x_2, \ldots\} \to \{0, 1\}$ erweitern, so dass $\beta \models \varphi$ gilt?

Hinweis

Zur Erinnerung: Zwei gerichtete Graphen $G = (V, E)$ und $G' = (V', E')$ heißen *isomorph*, wenn es eine bijektive Abbildung $f : V \to V'$ gibt, welche die Kantenrelation respektiert: für alle Paare (u, v) von Knoten aus G gilt $(u, v) \in E$ genau dann, wenn $\big(f(u), f(v)\big) \in E'$ gilt.

a) Zeigen Sie: GraphIsomorphismus $\in$ NP. Beschreiben Sie dazu, aus welchen Daten sich die Lösungskandidaten (die Zusatzeingabe) für eine Probleminstanz zusammensetzen und warum die Größe der Lösungskandidaten polynomiell durch die Größe der Instanz beschränkt ist. Beschreiben Sie anschließend, welche Eigenschaften von Eingabe und Zusatzeingabe ein Algorithmus überprüfen muss, um richtigerweise zu akzeptieren oder abzulehnen. Geben Sie für jeden Test eine obere (asymptotische) Laufzeitschranke an. **(3 Punkte)**

b) Zeigen Sie zunächst FinishSat $\leq_p$ Sat. Folgern Sie dann, dass FinishSat $\in$ NP gilt. **(3 Punkte)**

Lösung:

a) Wir nehmen im Folgenden an, dass die Graphen G und G' einer Eingabe-Instanz als Adjazenzmatrizen spezifiziert sind.

Lösungskandidaten. Ein Lösungskandidat zu einer Eingabe-Instanz (G, G') mit $G = (V, E)$ und $G' = (V', E')$ kann aus der Spezifikation einer Abbildung $f : V \to V'$ bestehen. Da der Definitionsbereich V endlich ist, kann dies beispielsweise in Form einer Liste $(v_1, v_1'), \ldots, (v_n, v_n')$ von Argument-Wert-Paaren geschehen (es gilt also $f(v_1) = v_1', \ldots, f(v_n) = v_n'$). Jedes der n Paare kann mit $2\log(n)$ Bits kodiert werden, wodurch sich eine Größenschranke von $O\big(n \log(n)\big) = O(n^2)$ für den Lösungskandidaten ergibt.

Algorithmus. Ein Algorithmus, der für einen wie oben beschriebenen Lösungskandida-

ten überprüft, ob er tatsächlich eine Lösung des GraphIsomorphismus-Problems zu einer Eingabe-Instanz (G, G') mit $G = (V, E)$ und $G' = (V', E')$ beschreibt, muss die Einhaltung der folgenden Eigenschaften testen:

Die Liste von Argument-Wert-Paaren spezifiziert eine totale Abbildung $f : V \to V'$. Alle „Argumente“ $v_1, \ldots, v_n$ müssen paarweise verschieden sein und es muss $V = \{v_1, \ldots, v_n\}$ gelten.
Dieser Test ist in quadratischer Zeit bezüglich der Größe des Lösungskandidaten und der Eingabe möglich.

Injektivität. Alle „Werte“ $v'_1, \ldots, v'_n$ müssen paarweise verschieden und in V' enthalten sein.
Dieser Test ist in quadratischer Zeit bezüglich der Größe des Lösungskandidaten und der Eingabe möglich.

Surjektivität. Jeder Wert $v' \in V'$ muss von der Abbildung angenommen werden, es muss also $V' = \{v'_1, \ldots, v'_n\}$ gelten.
Dieser Test ist in quadratischer Zeit bezüglich der Größe des Lösungskandidaten und der Eingabe möglich.

Kantenrespektierung. Für alle Kanten $(u, v) \in E$ muss auch $\big(u', v'\big) \in E'$ für $u' = f(u)$ und $v' = f(v)$ gelten. Umgekehrt muss für alle Kanten $(u', v') \in E'$ auch $\big(u, v\big) \in E$ für $u = f^{-1}(u')$ und $v = f^{-1}(v')$ gelten.
Das Auffinden der zu u, v gehörigen Werte u', v' ist in linearer Zeit möglich, der Enthaltenseinstest in der Kantenrelation E' ebenfalls. Die Anzahl solcher Iterationen ist durch die Anzahl der Kanten in E und somit ebenfalls linear beschränkt, sodass sich kombiniert eine quadratische Laufzeit ergibt. (Für die Gegenrichtung ergibt sich analog dieselbe Laufzeitschranke.)

b) Die Idee der Reduktion ist die Folgende: Wenn die partielle Wahrheitsbelegung α bereits ein Literal einer Klausel erfüllt, so erfüllt α bereits die ganze Klausel (unabhängig davon wie die Funktion vervollständigt wird). Solche Klauseln entfernen wir aus der gegebenen Formel φ. Außerdem entfernen wir alle Literale in den übrigen Klauseln, die unter α zu falsch auswerten. Die entstehende Formel φ' enthält dann nur noch Variablen, für die α nicht definiert ist, und nur noch („gekürzte“) Klauseln, die noch erfüllt werden müssen, indem Wahrheitswerte für diese Variablen bestimmt werden. Dabei sind zwei Spezialfälle zu beachten:

1. Aus einer Klauseln werden alle Literale entfernt, weil α alle Literale der Klausel mit falsch belegt. Dann kann keine Erweiterung von α die Formel erfüllen (weil die Klausel auch für jede Erweiterung zu falsch auswertet).
2. Alle Klauseln werden aus φ entfernt, weil α in jeder Klauseln ein Literal erfüllt. In diesem Fall erfüllt α die Formel bereits und die totale Erweiterung von α kann beliebig gewählt werden. Dazu nutzen wir die Konvention aus, dass eine Formel in KNF, die keine Klauseln enthält, immer zu wahr auswertet (Der Wahrheitswert wahr ist das neutrale Element der Konjunktion).

Formal definieren wir die Reduktion f durch folgenden Pseudo-Code-Algorithmus:

Algorithmus Reduktion f

Eingabe: Aussagenlogische Formel φ in KNF, partielle Wahrheitsbelegung α
Ausgabe: Aussagenlogische Formel in KNF

```
1:  for all Klauseln K aus φ do
2:    for all (positives) Literal x in K do
3:      if α(x) = 1 then
4:        Entferne die Klausel K aus φ und setze den Algorithmus mit der nächsten
          Klausel fort.
5:      if α(x) = 0 then
6:        Entferne das Literal x aus der Klausel.
7:        if K enthält keine Variablen mehr then
8:          return (irgendeine) eine unerfüllbare aussagenlogische Formel in KNF
9:    for all (negatives) Literal ¬x in K do
10:     if α(x) = 0 then
11:       Entferne die Klausel K aus φ und setze den Algorithmus mit der nächsten
          Klausel fort.
12:     if α(x) = 1 then
13:       Entferne das Literal ¬x aus der Klausel.
14:       if K enthält keine Variablen mehr then
15:         return (irgendeine) eine unerfüllbare aussagenlogische Formel in KNF
16: return die modifizierte Formel φ
```

Der Pseudo-Code-Algorithmus belegt, dass f total und berechenbar ist, denn für jede mögliche Eingabe liefert der Algorithmus eine Ausgabe. Der Funktionswert (und ob er überhaupt definiert ist) von α für eine Variable kann in polynomieller Zeit „nachgeschlagen“ werden. Außerdem iterieren alle Schleifen über die Klauseln und Literale der gegebenen Formel; die Anzahl der Schleifendurchläufe ist damit ebenfalls polynomiell in der Eingabelänge beschränkt. Zuletzt können Teile der Eingabe (Klauseln und Literale aus φ) auch in polynomieller Zeit gelöscht werden. Insgesamt ist f also in polynomieller Zeit berechenbar.

Es bleibt die Reduktionseigenschaft zu zeigen. Sei dazu $(\varphi, \alpha) \in$ FinishSat. Wir müssen zeigen, dass $\varphi' = f(\varphi, \alpha) \in$ Sat gilt. Wegen $(\varphi, \alpha) \in$ FinishSat gibt es eine (totale) Wahrheitsbelegung β, die φ erfüllt und α erweitert, d.h. es gilt $\alpha(x) = \beta(x)$ für alle x mit $\alpha(x) \neq \bot$. Wir können folgern, dass φ' aus φ durch das Entfernen von Klauseln und Literalen hervorgegangen ist und nicht eine in den Zeilen 8 oder 15 zurückgegebene unerfüllbare Formel ist: sonst wäre, wie zu Beginn argumentiert, φ nicht durch eine Erweiterung von α erfüllbar und die Existenz von β wäre ein Widerspruch. Sei K' eine Klausel aus φ' und K *eine* Klausel aus φ aus der K' durch das Löschen von Literalen durch f hervorgegangen ist. Die Klausel K wird von β erfüllt, weil β gerade ganz φ erfüllt. D.h. ein Literal x oder $\neg x$ aus K wird von β erfüllt. Dieses Literal kommt auch in K' vor, denn es kann keines der entfernten Literale sein. Es werden nämlich nur solche Literale entfernt, die von α (und damit von der Erweiterung β) nicht erfüllt werden. Also erfüllt β auch die Klausel K' und, da K' beliebig gewählt war, auch φ'. Es folgt, das $\varphi' \in$ Sat gilt.

Sei nun $\varphi' = f(\varphi, \alpha) \in$ Sat für eine Formel φ in KNF und eine partielle Wahrheitsbele-

gung α. Es ist zu zeigen, dass $(\varphi, \alpha) \in$ FinishSat gilt. Sei β die (totale) Wahrheitsbelegung, die φ' erfüllt und damit bezeugt, dass $\varphi' \in$ Sat gilt. Wir definieren die (totale) Wahrheitsbelegung γ für alle Variablen, die in φ vorkommen, wie folgt:

$$\gamma(x) = \begin{cases} \alpha(x) & \alpha(x) \neq \bot \\ \beta(x) & x \text{ kommt in } \varphi' \text{ vor} \\ 1 & \text{sonst} \end{cases}$$

Da φ' aus φ hervorgeht indem (unter anderem) alle Literale entfernt werden, für die α einen Wahrheitswert definiert (und damit alle Variablen für die α einen Wahrheitswert definiert) tritt für jede Variable genau einer der drei Fälle in der Definition von γ ein. Insbesondere belegt γ alle Variablen, die in φ' vorkommen genauso wie β, und alle Variablen, für die α definiert ist, genauso wie α. Also erfüllt γ alle Literale, die von α oder β erfüllt werden. Außerdem ist γ eine (totale) Erweiterung von α. Es bleibt damit zu zeigen, dass γ die Formel φ erfüllt. Sei dazu K eine Klausel aus φ. Wenn α ein Literal aus K auf wahr setzt, so erfüllt auch γ dieses Literal. Also erfüllt γ in diesem Fall die ganze Klausel. Gibt es ein solches Literal nicht, so gibt es in der Formel φ' eine Klausel K', die aus K durch das Entfernen von Literalen hervorgegangen ist. Da β die Formel φ' erfüllt und damit auch die Klausel K', erfüllt β ein Literal aus K'. Dann erfüllt auch γ dieses Literal und damit die Klausel K, denn alle Literale aus K' kommen auch in K vor. Also erfüllt γ alle Klauseln aus φ und damit φ selbst.

Insgesamt haben wir damit gezeigt, dass FinishSat $\leq_p$ Sat gilt. Nach Proposition 18.3 und, da nach Vorlesung Sat $\in$ NP gilt, folgt FinishSat $\in$ NP.

Alternative Lösung/alternative Reduktionsfunktion. Sei φ eine aussagenlogische Formel in KNF und $\alpha : \{x_1, x_2, \ldots\} \rightharpoonup \{0, 1\}$ eine partielle Wahrheitsbelegung. Die Idee der Reduktion ist Folgende: Wenn $\varphi \in$ Sat gilt, dann gibt es eine Wahrheitsbelegung β, so dass $\beta \models \varphi$ gilt. Um FinishSat zu entscheiden, sind wir aber daran interessiert, ob es eine Wahrheitsbelegung β gibt, so dass $\beta \models \varphi$ gilt *und* β mit α auf dem Definitionsbereich von α übereinstimmt, d.h. es gilt $\beta(x_i) = \alpha(x_i)$ für alle x_i mit $\alpha(x_i) \neq \bot$. Die Reduktion erweitert daher φ um solche Klauseln, die sicherstellen, dass nur solche Wahrheitsbelegungen die enstehende Formel erfüllen, die mit α auf dessen Definitionsbereich übereinstimmen.

Sei $\psi_\alpha = \bigwedge_{\alpha(x_i)=1} x_i \wedge \bigwedge_{\alpha(x_i)=0} \neg x_i$. Die Formel ψ_α enthält also die Klausel x_i, wenn α der Variablen x_i den Wert 1 zuweist und die Klausel $\neg x_i$, wenn α der Variablen x_i den Wert 0 zuweist. (Hierbei ist zu beachten, dass die Literale x_i und $\neg x_i$ bereits Klauseln sind.) Schließlich definieren wir die Reduktionsfunktion f, indem wir $f(\varphi, \alpha) = \varphi \wedge \psi_\alpha$ setzen.

Im Folgenden nehmen wir o.B.d.A. an, dass α durch eine Wertetabelle gegeben ist. Wir zeigen nun, dass es sich bei f tatsächlich um eine polynomielle Reduktion von FinishSat auf Sat handelt.

Totalität. An die Formel φ setzt die Funktion f keine Bedingungen. Die Formel ψ kann offensichtlich für jede partielle Wahrheitsbelegung generiert werden. Außerdem ist die entstehende Formel $\varphi \wedge \psi_\alpha$ wieder in KNF, also eine valide Eingabe für das Problem Sat. Die Funktion f ist also total.

Berechenbarkeit. Die Formel ψ_α kann berechnet (und an φ) angehängt werden, indem über die Wertetabelle von α iteriert wird, um die Klauseln x_i für Variablen mit $\alpha(x_i) = 1$ zu bestimmen und ein weiteres mal, um die Klauseln $\neg x_i$ für Variablen mit $\alpha(x_i) = 0$ zu bestimmen. Beide Durchläufe können in Linearzeit realisiert werden. Die Funktion f kann also in polynomieller Zeit berechnet werden.

Reduktionseigenschaft. Gelte $(\varphi, \alpha) \in$ FinishSat. Dann kann α zu einer (totalen) Wahrheitsbelegung β erweitert werden, so dass $\beta \models \varphi$ gilt. Wir zeigen nun, dass auch $\beta \models \psi_\alpha$ gilt. Enthalte ψ_α die Klausel x_i. Dies ist nach Konstruktion nur dann der Fall, wenn $\alpha(x_i) = 1$ gilt. Da β eine Erweiterung von α ist, gilt insbesondere $\beta(x_i) = \alpha(x_i) = 1$. Die Wahrheitsbelegung β ordnet der Klausel x_i also den Wert 1 zu.

Analog lässt sich zeigen, dass β einer Klausel $\neg x_i$ in ψ_α den Wert 1 zuweist (weil sie x_i den Wert 0 zuweist). Also erhält jede Klausel in ψ_α von β den Wert 1 und damit erhält auch ψ_α selbst den Wert 1.

Damit ist gezeigt, dass sowohl $\beta \models \varphi$ als auch $\beta \models \psi_\alpha$ gilt. Also gilt auch $\beta \models f(\varphi, \alpha) = \varphi \wedge \psi_\alpha$. Es folgt, dass $f(\varphi, \alpha) \in$ Sat gilt.

Für die Rückrichtung nehmen wir nun an, dass $f(\varphi, \alpha) \in$ Sat gilt. Dann gibt es eine (totale) Wahrheitsbelegung β mit $\beta \models \varphi \wedge \psi_\alpha$. Insbesondere gilt also $\beta \models \varphi$. Es genügt nun zu zeigen, dass β die partielle Wahrheitsbelegung α erweitert, also $\beta(x_i) = \alpha(x_i)$ für alle x_i mit $\alpha(x_i) \neq \bot$ gilt. Dann folgt sofort $(\varphi, \alpha) \in$ FinishSat.

Sei x_i eine Variable mit $\alpha(x_i) \neq \bot$. Angenommen es gilt $\alpha(x_i) = 0$. Dann enthält ψ_α die Klausel $\neg x_i$. Da β die Formel ψ_α (und damit alle ihre Klauseln) erfüllt, muss $\beta(x_i) = 0$ gelten, denn sonst würde β der Klausel $\neg x_i$ nicht den Wert 1 zuordnen. Es gilt also $\beta(x_i) = \alpha(x_i) = 0$. Analog lässt sich in dem anderen Fall $\alpha(x_i) = 1$ zeigen, dass ebenfalls $\beta(x_i) = 1$ gilt.

Damit ist die Reduktionseigenschaft gezeigt:

$$(\varphi, \alpha) \in \text{FinishSat} \Leftrightarrow f(\varphi, \alpha) \in \text{Sat}.$$

Insgesamt haben wir damit gezeigt, dass FinishSat $\leq_p$ Sat gilt. Nach Proposition 18.3 und, da nach Vorlesung Sat $\in$ NP gilt, folgt FinishSat $\in$ NP.

Aufgabe 12.2 [Reduktionseigenschaften überprüfen] **9 Punkte**

Wir betrachten die beiden Entscheidungsprobleme ExactCover und 0-1-ILP, welche nachfolgend angegeben sind.

Problem: ExactCover
Gegeben: Endliche Menge M und Teilmengen $S_1, \ldots, S_k \subseteq M$
Frage: Gibt es eine Indexmenge $I \subseteq \{1, \ldots, k\}$, sodass M die *disjunkte* Vereinigung der Mengen S_i mit $i \in I$ ist, d.h. gilt $M = \bigcup_{i \in I} S_i$ und $S_i \cap S_j = \emptyset$ für alle $i \neq j$ in I?

Beispiel

Wir betrachten die Menge $M = \{a, b, c, d\}$ und die folgenden Teilmengen von M.

$$\begin{aligned} S_1 &= \{a, b\} \\ S_2 &= \{c, d\} \\ S_3 &= \{b, c, d\} \end{aligned}$$

Die Indexmenge $I = \{1, 2\}$ belegt, dass $(M, (S_1, S_2, S_3))$ in ExactCover ist, weil $M = S_1 \cup S_2$ und $S_1 \cap S_2 = \emptyset$ gilt. Die Indexmenge $J = \{1, 3\}$ tut dies allerdings nicht. Es gilt zwar $M = S_1 \cup S_3$, aber S_1 und S_3 sind nicht disjunkt, es gilt $S_1 \cap S_3 = \{b\} \neq \emptyset$.

Problem: 0-1-ILP
Gegeben: Ein System $\mathcal{U} = (U_1, \ldots, U_m)$ ganzzahliger linearer Ungleichungen über einer endlichen Variablenmenge X.
Frage: Gibt es eine Wertebelegung $\beta : X \to \{0, 1\}$ der Variablen, die alle Ungleichungen erfüllt?

Beispiel

Für das folgende System von Ungleichungen ist $\beta = \{x \mapsto 1,\ y \mapsto 0,\ z \mapsto 1\}$ eine erfüllende Wertebelegung (auch *Lösung* genannt).

$$\begin{array}{rcrcrl} 4x & - & 2y & + & z & \geq 3 \\ & & 11y & - & 2z & \leq 8 \end{array}$$

Betrachten Sie die folgende Reduktion f von ExactCover auf 0-1-ILP und untersuchen Sie ihre Eigenschaften, indem Sie die Teilaufgaben lösen.

Funktion f

Einer Menge M und Teilmengen $S_1, \ldots, S_k \subseteq M$ wird durch die Reduktion f das folgende System von Ungleichungen über den Indikatorvariablen x_i für $i \in \{1, \ldots, k\}$ zugewiesen. Dabei soll eine Variable x_i intuitiv genau dann den Wert 1 annehmen, wenn i zur Indexmenge I gehört, und andernfalls den Wert 0.

$$\begin{array}{ll} c_{1,e}x_1 + \cdots + c_{k,e}x_k \geq 1 & \text{für alle } e \in M \\ x_i + x_j \leq 1 & \text{für alle } i \neq j \text{ mit } S_i \cap S_j \neq \emptyset \end{array}$$

wobei $c_{i,e}$ entweder für die Konstante 1 steht, wenn $e \in S_i$ gilt, oder für die Konstante 0, falls $e \notin S_i$ gilt.

a) Betrachten Sie die Menge $M = \{a, b, c, d, h\}$ und die folgenden Teilmengen von M.

$$\begin{aligned} S_1 &= \{a, b\} \\ S_2 &= \{b, c\} \\ S_3 &= \{d, h\} \\ S_4 &= \{a\} \end{aligned}$$

Gilt $(M, (S_1, S_2, S_3, S_4)) \in$ ExactCover? Falls ja, geben Sie eine entsprechende Indexmenge I an, die dies bezeugt. **(0,5 Punkte)**

b) Stellen Sie das Ungleichungssytem $f(M, (S_1, \ldots, S_4))$ für die Menge M und die Teilmengen S_1, S_2, S_3, S_4 aus Teilaufgabe a) auf. Ist das System in 0-1-ILP? Wenn ja, geben Sie eine erfüllende Wertebelegung $\beta : \{x_1, \ldots, x_4\} \to \{0, 1\}$ für das System an. **(1,5 Punkte)**

c) Beschreiben Sie die Bedeutung der erzeugten Ungleichungen. Welchen Eigenschaften der Teilmengen S_i mit $i \in I$ entsprechen sie? **(1 Punkt)**

d) Beweisen Sie, dass die angegebene Funktion die Reduktionseigenschaft besitzt. **(4,5 Punkte)**

e) Begründen Sie, dass die Reduktionsfunktion in polynomieller Zeit berechnet werden kann. Geben Sie dazu insbesondere eine asymptotische obere Laufzeitschranke an. **(1,5 Punkte)**

Lösung:

a) Ja, es gilt $(M, (S_1, S_2, S_3, S_4)) \in$ ExactCover. Eine bezeugende Indexmenge ist $I = \{2, 3, 4\}$. Es gilt nämlich $S_2 \cup S_3 \cup S_4 = \{b, c, d, h, a\} = M$ und $S_2 \cap S_3 = \emptyset$, $S_2 \cap S_4 = \emptyset$ sowie $S_3 \cap S_4 = \emptyset$.

b) Wir geben zunächst das Ungleichungssytem an.

$$\begin{aligned} x_1 + x_4 &\geq 1 && \text{für } a \in M \\ x_1 + x_2 &\geq 1 && \text{für } b \in M \\ x_2 &\geq 1 && \text{für } c \in M \\ x_3 &\geq 1 && \text{für } d \in M \text{ und } h \in M \\ x_1 + x_2 &\leq 1 && \text{wegen } S_1 \cap S_2 = \{b\} \neq \emptyset \\ x_1 + x_4 &\leq 1 && \text{wegen } S_1 \cap S_4 = \{a\} \neq \emptyset \\ x_2 + x_1 &\leq 1 && \text{wegen } S_2 \cap S_1 = \{b\} \neq \emptyset \\ x_4 + x_1 &\leq 1 && \text{wegen } S_4 \cap S_1 = \{a\} \neq \emptyset \end{aligned}$$

Die Wertebelegung $\beta = \{x_1 \mapsto 0,\ x_2 \mapsto 1,\ x_3 \mapsto 1,\ x_4 \mapsto 1\}$ belegt, dass das System in 0-1-ILP ist, denn sie erfüllt alle Ungleichungen.

c) Die Ungleichungen

$$c_{1,e}x_1 + \ldots + c_{k,e}x_k \geq 1 \quad \text{für alle } e \in M$$

repräsentieren die Forderung, dass jedes Element aus M in einem der ausgewählten S_i enthalten ist; und M damit die Vereinigung der ausgewählten S_i ist. (Um die Ungleichung zu erfüllen muss nach Definition der $c_{i,e}$ ein x_i auf 1 gesetzt werden, sodass $e \in S_i$ gilt.)

Die Ungleichungen

$$x_i + x_j \leq 1 \text{ für alle } i \neq j \text{ mit } S_i \cap S_j \neq S_i$$

repräsentieren die Forderung, dass die „ausgewählten“ S_i schnittfrei sind. Setzt eine Wertebelegung zwei Variablen x_i und x_j mit $i \neq j$ auf den Wert 1, aber es gilt $S_i \cap S_j \neq \emptyset$, so würde die entsprechende Ungleichung $x_i \neq x_j \leq 1$, wie intendiert, nicht durch die Wertebelegung erfüllt werden.

d) Zu zeigen ist, dass eine Eingabe $(M, (S_1, \ldots, S_k))$ genau dann in ExactCover ist, wenn das zugehörige Ungleichungssystem $f(M, (S_1, \ldots, S_k))$ eine Lösung hat.

Sei $(M, (S_1, \ldots, S_k)) \in$ ExactCover und $I \subseteq \{1, \ldots, k\}$ eine Indexmenge, die dies belegt. Das heißt, es gilt $M = \bigcup_{i \in I} S_i$ und $S_i \cap S_j = \emptyset$ für alle $i \neq j$ mit $i, j \in I$. Wir konstruieren aus I eine Wertebelegung β für das Ungleichungssystem $\mathcal{U} = f(M, (S_1, \ldots, S_k))$, indem wir

$$\beta(x_i) = \begin{cases} 1, & \text{falls } i \in I \\ 0, & \text{sonst} \end{cases}$$

setzen. Wir müssen zeigen, dass β das Ungleichungssytem erfüllt. Sei dazu $e \in M$ beliebig und $i \in I$ der *eindeutige* Index in I, sodass $e \in S_i$ gilt. Eine solches i existiert, da M die Vereinigung aller S_i mit $i \in I$ ist. Dann gilt nach Konstruktion das $c_{i,e} = 1$ und $\beta(x_i) = 1$. Es folgt damit (und der Tatsache, das alle Konstanten entweder 0 oder 1 sind und der Wertebereich von β auch nur die Werte 0 und 1 beinhalten kann), dass

$$c_{1,e}\beta(x_1) + \ldots + c_{k,e}\beta(x_k) \geq c_{i,e}\beta x_i = 1 \cdot 1 \geq 1$$

gilt. Da das Element e beliebig gewählt war, erfüllt β alle Ungleichungen des Typs $c_{1,e}x_1 + \ldots + c_{k,e}x_k \geq 1$.

Seien S_i, S_j Teilmengen mit $i \neq j$ und $S_i \cap S_j \neq \emptyset$. Wir müssen zeigen, dass β die Ungleichung $x_i + x_j \leq 1$ erfüllt. Da die Indexmenge I bezeugt, dass $(M, (S_1, \ldots, S_k)) \in$ ExactCover gilt, gilt entweder $i \notin I$ oder $j \notin I$. Also folgt $\beta(x_i) = 0$ oder $\beta(x_j) = 0$. In beiden Fällen folgt sofort, dass $\beta(x_i) + \beta(x_j) \leq 1$ gilt.

Insgesamt erfüllt β also alle Ungleichungen und damit ist das System $\mathcal{U}$ in 0-1-ILP.

Sei nun umgekehrt ein Ungleichungssystem $f(M, (S_1, \ldots, S_k))$ mit erfüllender Wertebelegung β zu einer Instanz $(M, (S_1, \ldots, S_k))$ gegeben.

Wir setzen $I = \{i \mid \beta(x_i) = 1\}$. Es genügt zu zeigen, dass M die Vereinigung aller S_i mit $i \in I$ ist und $S_i \cap S_j = \emptyset$ für alle $i, j \in I$, $i \neq j$ gilt.

Wir zeigen zunächst, dass M die Vereinigung aller S_i mit $i \in I$ ist. Sei $e \in M$ beliebig. Nach der Ungleichung

$$c_{1,e}x_1 + \ldots + c_{k,e}x_k \geq 1$$

muss es ein i geben, sodass $c_{i,e} = 1$ und $\beta(x_i) = 1$ gilt, da die Konstanten und Variablenbelegungen durch β alle entweder 0 oder 1 sind. Dann gilt nach Definition von I gerade $i \in I$. Außerdem gilt nach Konstruktion $e \in S_i$, weil $c_{i,e} = 1$ gilt. Insbesondere ist e also in der Vereinigung aller S_i mit $i \in I$ enthalten.

Es bleibt zu zeigen, dass $S_i \cap S_j = \emptyset$ für alle $i, j \in I$, $i \neq j$ gilt. Dazu nehmen wir an, dass dies nicht der Fall ist und leiten einen Widerspruch dazu her, dass β das Ungleichungssystem erfüllt. Angenommen es gilt $S_i \cap S_j \neq \emptyset$ für $i, j \in I$ und $i \neq j$. Da sowohl i als auch j in I sind, gilt nach Definition von I gerade $\beta(x_i) = \beta(x_j) = 1$. Außerdem ist die Ungleichung $x_i + x_j \leq 1$ im System enthalten, weil $S_i \cap S_j \neq \emptyset$ gilt. Aber dies ist ein Widerspruch, denn $\beta(x_i) + \beta(x_j) = 1 + 1 = 2 > 1$, β erfüllt die Ungleichung $x_i + x_j \leq 1$ also *nicht*. Es muss also $S_i \cap S_j = \emptyset$ für alle $i, j \in I$ mit $i \neq j$ gelten.

Damit ist gezeigt, dass $(M, (S_1, \ldots, S_k)) \in$ ExactCover gilt.

e) Wir können annehmen, dass jedes Element aus M und jede Teilmenge S_i mit mindestens einem Bit kodiert sind. Sei n die Anzahl der Elemente in M. Dann ist die Eingabelänge mindestens $n + k$ (zur Erinnerung: k ist die Anzahl der gegebenen Teilmengen).

Die Variablen können in Linearzeit berechnet werden, d.h. $\mathcal{O}(n + k)$, indem über die Eingabe gelaufen wird und für jede angetroffene Teilmenge die entsprechende Variable bestimmt wird. Ebenso können die Konstanten $c_{i,e}$ bestimmt (und gespeichert) werden, indem für jedes Element in M über die Eingabe gelaufen wird und die Zugehörigkeit zu jedem S_i getestet wird. Dies ist mit einem Aufand von $\mathcal{O}((n + k)^2)$ möglich.

Um die Ungleichungen des Types $c_{1,e}x_i + \ldots + c_{k,e}x_k$ zu berechnen kann verschachtelt über die Elemente aus M, die zuvor berechneten Variablen und die zuvor berechneten Konstanten iteriert werden. Insgesamt ergibt sich dafür ein Aufwand von $\mathcal{O}((n + k)^4)$.

Die Ungleichungen des Types $x_i + x_j \leq 1$ können berechnet werden indem verschachtelt über alle Paare von Teilmengen S_i in der Eingabe iteriert wird. Für jedes Paar S_i, S_j mit $i \neq j$ muss dann geprüft werden, ob $S_i \cap S_j \neq \emptyset$ gilt, und, falls ja, die Ungleichung $x_i + x_j \leq 1$ aufgestellt werden. Dafür müssen maximal $2n$ Elemente verglichen werden; dies ist mit einem quadratischen Aufwand möglich. Mit bekanntem i und j ist das Aufstellen der Ungleichung dann in Linearzeit möglich (genau gesagt, können die Kodierungen von i und j höchstens Länge $\log(k)$ annehmen). Insgesamt ergibt sich ein Aufwand von $\mathcal{O}((n+k)^4)$ für das Bestimmen der Ungleichungen.

Wir folgern, dass f in polynomieller Zeit berechenbar ist; der Aufwand ist nach obiger Beschreibung nämlich ingesamt $\mathcal{O}((n + k)^4)$, und $n + k$ ist durch die Eingabelänge beschränkt.

Zusatzaufgabe [Sudoku]

In Kapitel 18 der Vorlesung wurden Sudoku-Rätsel als Beispiel für P vs. NP betrachtet. Wir betrachten in dieser Aufgabe die Folgende Verallgemeinerung von Sudoku-Rätseln. Für $k \in \mathbb{N}$ soll das Spielfeld die Größe $k^2 \times k^2$ haben. Alle Boxen bestehen dann aus $k \times k$ Feldern und sollen so mit den Zahlen 1 bis k^2 gefüllt werden, dass keine Box, keine Spalte und keine Reihe zwei gleiche Zahlen enthält. Wie beim gewöhnlichen Sudoku sind die Zahlen für einige Positionen des Spielfeldes schon vorgegeben. Das allgemeine Sudoku-Rätsel ist also wie folgt definiert.

Problem:	Sudoku
Gegeben:	1^k für $k \in \mathbb{N}$, eine partielle Funktion $f : \{1, 2, \ldots, k^2\}^2 \rightharpoonup \{1, 2, \ldots, k^2\}$
Frage:	Besitzt das durch k und f gegebene Sudoku-Rätsel eine Lösung?

Hierbei beschreibt k die Größe des Spielfeldes und f gibt an, ob und mit welcher Zahl eine Position bereits belegt ist. Ist f für eine Position i nicht definiert, d.h. $f(i) = \perp$, so ist die Position i *nicht* belegt, andernfalls ist sie mit der Zahl $f(i)$ belegt. Die herkömmlichen Sudoku-Rätsel sind also der Spezialfall $k = 3$ des allgemeinen Sudoku-Rätsels.

a) Lässt sich das Sudoku-Rätsel für den Fall $k = 23$ in polynomieller Zeit lösen?

b) Bei der folgenden Variante des Färbungsproblems muss eine Teilfärbung zu einer Färbung vervollständigt werden.

Problem: FinishCol
Gegeben: Ungerichteter Graph $G = (V, E)$, Zahl k, partielle Funktion $c : V \rightharpoonup \{1, \ldots, k\}$
Frage: Lässt sich c zu einer totalen Funktion (auf V) erweitern, die eine zulässige Färbung von G (mit k Farben) ist?

Für die Knoten, für die c definiert ist, sind die Farben also bereits festgelegt.

Geben Sie eine polynomielle Reduktion von Sudoku auf FinishCol an und beweisen Sie deren Korrektheit.

Hinweis: Fassen Sie die Positionen des Spielfeldes als Knoten eines Graphen auf.

c) Geben Sie eine polynomielle Reduktion von FinishCol auf Col an und beweisen Sie deren Korrektheit.

Hinweis: Bilden Sie einen gegebenen Graphen G auf einen Graphen G' mit k zusätzlichen Knoten ab. Welche zusätzlichen Kanten muss G' haben?