ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
http://www.ita.inf.ethz.ch/theoInf18

Lösungsvorschläge – Blatt 10

Zürich, 30. November 2018

Lösung zu Aufgabe 28

(a) Sei M eine nichtdeterministische MTM mit $\text{Time}_M(n) \in O(n^2)$ und die ausserdem die Platzschranke $O(n)$ auf jeder Berechnung einhält. Um zu zeigen, dass $L(M) \in \text{SPACE}(n \log n)$ gilt, konstruieren wir eine deterministische MTM A mit $L(A) = L(M)$, die die Berechnungen von M simuliert. Für die Konstruktion orientieren wir uns am Beweis des Satzes von Savitch aus dem Buch. Wir nehmen also wieder an, dass M für alle Wörter $w \in L(M)$ nur eine eindeutige akzeptierende Konfiguration besitzt, so dass die deterministische Simulation der Arbeit von M, die A durchführt, nur feststellen muss, ob die akzeptierende Konfiguration $C_{\text{accept}}(w)$ von der Startkonfiguration C_{start} aus erreichbar ist. Wegen der Voraussetzung über die Zeitkomplexität von M kann eine kürzeste akzeptierende Berechnung von M auf w höchstens die Länge $d \cdot |w|^2$ haben für eine geeignete Konstante d. Um festzustellen, ob $C_{\text{accept}}(w)$ aus $C_{\text{start}}(w)$ in $d \cdot |w|^2$ Schritten erreichbar ist, verwenden wir dieselbe Prozedur REACHABLE wie im Beweis des Satzes von Savitch. Weil die Funktion $\log_2(d \cdot n^2) = 2 \log_2 n + \log_2 d$ offenbar platzkonstruierbar ist, kann A den Wert $d \cdot |w|^2$ für jedes beliebige Wort mit $2 \log_2 |w| + \log_2 d$ Speicherplatz ausrechnen und abspeichern. Jede innere Konfiguration einer Berechnung von M auf w kann in $c \cdot |w|$ Platz gespeichert werden, weil nach Voraussetzung jede Berechnung mit $O(n)$ Platz auskommt. Bei der Durchführung von REACHABLE müssen höchstens $O(\log_2 |w|)$ Konfigurationen auf einmal gespeichert werden, weil die Rekursionstiefe logarithmisch in der Zeitkomplexität von M ist. Damit ergibt sich mit derselben Argumentation wie im Beweis des Satzes von Savitch ein Platzbedarf in $O(|w| \cdot \log |w|)$ für A.

(b) Eine ähnliche Konstruktion wie im Aufgabenteil (a) oder im Beweis des Satzes von Savitch funktioniert hier nicht. Für jede Sprache L in $\text{NSPACE}(f(n)) \cap \text{NTIME}(f(n)^k)$ gilt $L \in \text{NSPACE}(f(n))$ und $L \in \text{NTIME}(f(n)^k)$. Also gibt es eine nichtdeterministische MTM M_1 mit $L(M_1) = L$ und $\text{Space}_{M_1}(n) \in O(f(n))$ und eine nichtdeterministische MTM M_2 mit $L(M_2) = L$ und $\text{Time}_{M_2}(n) \in O((f(n))^k)$. Es kann also sein, dass es eine MTM gibt, die L mit kleiner Platzkomplexität entscheidet, und eine andere MTM, die L mit geringer Zeitkomplexität entscheidet. Um einen Beweis wie in Aufgabenteil (a) führen zu können, brauchen wir aber *eine* nichtdeterministische MTM, die beide Schranken für Platz und Zeit einhält.

Lösung zu Aufgabe 29

(a) Sei ϕ eine Eingabe für 3SAT, also eine 3KNF-Formel $\phi = C_1 \wedge \ldots \wedge C_m$ mit den Klauseln $C_1, \ldots, C_m$ über den Variablen aus $X = \{x_1, \ldots, x_n\}$. Wir wollen ϕ in eine Eingabe ψ für E3SAT verwandeln. Hierfür modifizieren wir ϕ zunächst so, dass in keiner Klausel eine Variable mehrfach auftaucht. Dazu streichen wir allfällige Wiederholungen eines Literals innerhalb einer Klausel ersatzlos. An der Erfüllbarkeit ändert sich dadurch nichts. Enthält eine Klausel zugleich eine Variable y und ihre Negation $\overline{y}$ als Literal, so ist die gesamte Klausel für jede beliebige Belegung erfüllt. Solche Klauseln können wir also ersatzlos aus ϕ streichen, ohne die Erfüllbarkeit zu beeinflussen. Insgesamt dürfen wir daher ohne Beschränkung der Allgemeinheit davon ausgehen, dass in jeder Klausel von ϕ jede Variable höchstens einmal vorkommt. Die Literale einer Klausel gehören dann immer zu paarweise unterschiedlichen Variablen.

Wir können ψ jetzt wie folgt konstruieren: Alle Klauseln von ϕ, die bereits drei Literale enthalten, bleiben unverändert. Eine Klausel $C_i = (l_{i,1} \vee l_{i,2})$ wird ersetzt durch die zwei neuen Klauseln $C_{i,1} = (l_{i,1} \vee l_{i,2} \vee y_i)$ und $C_{i,2} = (l_{i,1} \vee l_{i,2} \vee \overline{y_i})$, wobei y_i eine neue Variable ist, die sonst nirgends in ψ vorkommt. Eine Klausel $C_i = (l_i)$ wird ersetzt durch die vier neuen Klauseln $C_{i,1} = (l_i \vee y_{i,1} \vee y_{i,2})$, $C_{i,2} = (l_i \vee y_{i,1} \vee \overline{y_{i,2}})$, $C_{i,3} = (l_i \vee \overline{y_{i,1}} \vee y_{i,2})$ und $C_{i,4} = (l_i \vee \overline{y_{i,1}} \vee \overline{y_{i,2}})$, wobei $y_{i,1}$ und $y_{i,2}$ zwei neue Variablen sind, die sonst nirgends in ψ vorkommen. Offenbar ist diese Konstruktion in polynomieller Zeit durchführbar und die Zusatzbedingungen von E3SAT gegenüber 3SAT sind erfüllt.

Wir zeigen nun, dass ϕ genau dann erfüllbar ist, wenn ψ erfüllbar ist.

Sei α eine erfüllende Belegung für ϕ. Dann setzt α mindestens ein Literal in jeder der Klauseln $C_1, \ldots, C_m$ auf 1. Damit ist aber durch eine beliebige Erweiterung von α auch jede Klausel von ψ erfüllt, weil diese Klauseln höchstens noch zusätzliche Literale enthalten.

Sei β eine erfüllende Belegung für ψ. Dann werden alle Klauseln in ψ von β erfüllt. Wir wollen zeigen, dass die Einschränkung von β auf die Variablen aus X dann auch ϕ erfüllt. Jede Klausel aus ϕ der Länge 3 wurde unverändert in ψ übernommen und wird offenbar von β erfüllt. Für eine Klausel $C_i = (l_{i,1} \vee l_{i,2})$ der Länge 2 in ϕ gibt es in ψ die beiden Klauseln $C_{i,1}$ und $C_{i,2}$. Die Variable y_i kommt in einer dieser beiden Klauseln positiv vor und in der anderen Klausel negativ. Wir nehmen zunächst an, dass $\beta(y_i) = 1$. Da nach Voraussetzung jede Klausel in ψ von β erfüllt wird, ist auch $C_{i,2}$ erfüllt, obwohl das Literal $\overline{y_i}$ hier auf 0 gesetzt ist. Also muss $C_i = (l_{i,1} \vee l_{i,2})$ von β erfüllt werden. Der Fall, dass $\beta(y_i) = 0$ gilt, ist analog.

Für eine Klausel C_i der Länge 1 in ϕ gibt es vier Klauseln in ψ, so dass für jede Belegung der beiden neuen Variablen $y_{i,1}$ und $y_{i,2}$ eine dieser Klauseln nur erfüllt sein kann, wenn auch C_i erfüllt wird. Insgesamt erfüllt β also alle Klauseln von ϕ.

(b) Wir zeigen zunächst, dass LARGE-CLIQUE NP-vollständig ist. In der Vorlesung wurde bereits gezeigt, dass CLIQUE in NP enthalten ist. Weil jede LARGE-CLIQUE-Instanz insbesondere eine Instanz von CLIQUE ist, ist mit derselben Argumentation auch LARGE-CLIQUE in NP.

Um zu zeigen, dass LARGE-CLIQUE eine NP-schwere Sprache ist, zeigen wir E3SAT $\leq_p$ LARGE-CLIQUE. Wir verwenden hierfür exakt dieselbe Reduktion wie beim Beweis von SAT $\leq_p$ CLIQUE im Buch, um eine Formel in E3KNF in eine CLIQUE-Instanz (G, k) umzuwandeln. Für eine Formel ϕ mit m Klauseln

hat der konstruierte Graph $G = (V, E)$ genau $3m$ Knoten. Nach Konstruktion gilt $k = m = |V|/3$. Mit dem Beweis von Lemma 6.9 aus dem Buch folgt also unmittelbar die Behauptung.

Wir zeigen nun, dass VERY-LARGE-CLIQUE in P liegt. Für einen Graphen mit n Knoten können wir einfach alle $\binom{n}{n-3} + \binom{n}{n-2} + \binom{n}{n-1} + \binom{n}{n} \in O(n^3)$ möglichen Teilmengen von mindestens $n-3$ Knoten darauf überprüfen, ob diese eine Clique bilden. Diese Überprüfung ist offenbar für jede der Teilmengen in Zeit $O(n^2)$ möglich. Es ergibt sich also insgesamt eine polynomielle Laufzeit in $O(n^5)$.

Lösung zu Aufgabe 30

(a) Die Idee unserer Reduktion ist es, SCP als eine Verallgemeinerung von VC anzusehen.

Die Eingabe für VC ist ein Graph $G = (V, E)$ und eine natürliche Zahl k. Jede solche Eingabe bilden wir wie folgt auf eine Eingabe für SCP ab. Sei $E_v := \{e \in E \mid v \text{ ist inzident zu } e\}$ die Menge der mit dem Knoten v inzidenten Kanten und sei $\mathcal{S}_G = \{E_v \mid v \in V\}$. Dann ist

$$(E, \mathcal{S}_G, k)$$

unsere Eingabe für SCP. Die Mengenfamilie $\mathcal{S}_G$ enthält also für jeden Knoten v die Menge aller Kanten, die inzident mit v sind. Man beachte, dass zwei Knoten v und w dieselbe Menge $E_v = E_w$ inzidenter Kanten haben können; in diesem Fall ist E_v nur einmal in $\mathcal{S}$ enthalten.

Die Transformation ist offensichtlich in polynomieller Zeit durchführbar. Es bleibt noch zu zeigen, dass (G, k) genau dann eine zu akzeptierende Eingabe für VC ist, wenn $(E, \mathcal{S}_G, k)$ eine zu akzeptierende Eingabe für SCP ist.

Angenommen, es gibt ein Vertex-Cover der Grösse k in G. Dann gibt es eine Menge von k Knoten $\{v_1, v_2, \ldots, v_k\}$, die alle Kanten in G überdeckt. Wir können die überdeckten Kanten zudem durch $\bigcup_{i=1}^{k} E_{v_i}$ beschreiben. Jede Menge E_{v_i} ist in $\mathcal{S}_G$. Dementsprechend gibt es ein Set-Cover der Grösse höchstens k für $(E, \mathcal{S}_G, k)$.

Angenommen, es gibt ein Set-Cover der Grösse k für $(E, \mathcal{S}_G, k)$. Dann gibt es ein $\mathcal{C} \subseteq \mathcal{S}_G$, so dass $|\mathcal{C}| = k$ und $\bigcup_{S \in \mathcal{C}} S = E$ gilt. Damit ist die Menge der k Knoten, die in unserer Reduktion auf die Mengen aus $\mathcal{C}$ abgebildet werden, inzident mit allen Kanten aus E. Dementsprechend bilden diese Knoten ein Vertex-Cover der Grösse k in G. Falls mehrere Knoten dieselbe Menge inzidenter Kanten haben und deshalb auf dieselbe Menge aus $\mathcal{C}$ abgebildet werden, reicht es offenbar aus, für jede Menge aus $\mathcal{C}$ einen solchen Knoten für den Vertex-Cover auszuwählen.

(b) Sei $(X, \mathcal{S}, k)$ eine Eingabe für SCP mit $X = \{x_1, x_2, \ldots, x_n\}$ und $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ und einer natürlichen Zahl k. Jede solche Eingabe bilden wir wie folgt auf eine Eingabe (G, k) mit $G = (V, E)$ für DS ab. Für die Knoten von G gilt $V = V_X \uplus V_{\mathcal{S}}$ mit $V_X = \{x_1, x_2, \ldots, x_n\}$ und $V_{\mathcal{S}} = \{s_1, s_2, \ldots, s_m\}$, wobei V_X also den Elementen aus X entspricht und $V_{\mathcal{S}}$ den Mengen aus $\mathcal{S}$. Für die Kanten von G gilt zum einen, dass die Knoten aus $V_{\mathcal{S}}$ eine Clique bilden. Ferner ist

$$\{x_i, s_j\} \in E \iff x_i \in S_j \,,$$

d. h., eine Kante zwischen einem Knoten von V_X und einem Knoten von $V_{\mathcal{S}}$ gibt es genau dann, wenn das Element $x_i \in X$ in der Menge $S_j \in \mathcal{S}$ enthalten ist, das S_j das

Element x_i also überdeckt. Diese Konstruktion kann offensichtlich in polynomieller Zeit durchgeführt werden.

Sei nun $(X, \mathcal{S}, k)$ eine Eingabe für SCP, so dass X ein Set-Cover aus $\mathcal{S}$ der Grösse k besitzt; sei $\mathcal{C}$ ein solches Set-Cover. Die entsprechenden Knoten aus $V_\mathcal{S}$ sind ein Dominating-Set D derselben Grösse, was wie folgt begründet werden kann. Alle Knoten aus $V_\mathcal{S}$ sind trivialerweise dominiert, da $V_\mathcal{S}$ eine Clique ist. Ferner ist jeder Knoten aus V_X adjazent zu einem ausgewählten Knoten in $V_\mathcal{S}$, da jedes Element aus X in mindestens einer Menge aus $\mathcal{C}$ beinhaltet ist.

Sei andererseits D ein Dominating-Set der Grösse k von G. Falls $D \subseteq V_\mathcal{S}$, ist die entsprechende Auswahl an Mengen aus $\mathcal{S}$ ein Set-Cover der gleichen Grösse für X, was wie oben begründet werden kann. Falls ein Knoten $x \in D \setminus V_\mathcal{S}$ existiert, so können wir D modifizieren, indem wir x gegen einen adjazenten Knoten aus $V_\mathcal{S}$ austauschen. Die beiden Knoten bleiben dabei dominiert und kein anderer Knoten von V_X verliert mit x einen dominierenden Knoten, da sie untereinander nicht verbunden sind, während die Knoten aus $V_\mathcal{S}$ in jedem Fall dominiert sind (da $V_\mathcal{S}$ eine Clique ist).