ETH zürich
Departement Informatik

Theoretische Informatik

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

Lösungsvorschläge – Blatt 11

Zürich, 7. Dezember 2018

Lösung zu Aufgabe 31

Wir betrachten die in der Aufgabenstellung gegebene Reduktion und wollen zeigen, dass der konstruierte Graph G_ϕ genau dann ein Vertex-Cover der Grösse $5q$ hat, wenn die E3-KNF-Formel ϕ mit q Klauseln erfüllbar ist.

Sei Δ_s das Dreieck aus den Knoten $V_{s,1}$, $V_{s,2}$ und $V_{s,3}$ für $1 \leq s \leq q$. Sei H_i der Kreis in G_ϕ bestehend aus den Knoten $T_{i,1}, F_{i,1}, T_{i,2}, F_{i,2}, \ldots, T_{i,m(i)}, F_{i,m(i)}$ für $i \in \{1, \ldots, n\}$. Wir bemerken zunächst, dass jeder der Kreise H_i genau zwei Vertex-Cover der Grösse $m(i)$ hat, nämlich $\{T_{i,1}, T_{i,2}, \ldots, T_{i,m(i)}\}$ und $\{F_{i,1}, F_{i,2}, \ldots, F_{i,m(i)}\}$. Weiter gibt es kein Vertex-Cover für H_i, das weniger als $m(i)$ Knoten enthält.

Sei α eine erfüllende Belegung für ϕ. Wir konstruieren aus α ein Vertex-Cover S für G_ϕ wie folgt:

- Für jede Variable x_i mit $\alpha(x_i) = 1$ nehmen wir alle Knoten $T_{i,1}, T_{i,2}, \ldots, T_{i,m(i)}$ in das Vertex-Cover S auf,
- für jede Variable x_i mit $\alpha(x_i) = 0$ nehmen wir alle Knoten $F_{i,1}, F_{i,2}, \ldots, F_{i,m(i)}$ in S auf und
- für jede Klausel C_s nehmen wir genau zwei Knoten des Dreiecks Δ_s in S auf, so dass ein Knoten, der einem von α auf 1 gesetzten Literal entspricht, nicht genommen wird. Ein solcher Knoten existiert immer, weil α eine erfüllende Belegung ist.

Die Menge S enthält genau $2q$ Knoten aus den Klausel-Dreiecken Δ_s und genau $3q$ Knoten aus den Kreisen, weil die Kreise eine Gesamtlänge von $6q$ haben: Jede Klausel enthält genau drei Variablenvorkommen und für jedes Variablenvorkommen gibt es genau zwei Knoten in den Kreisen.

Die Knoten aus S überdecken alle Kanten der Kreise H_i und alle Kanten der Klausel-Dreiecke Δ_s. Es bleibt zu zeigen, dass sie auch alle Kanten aus den Mengen E_1 und E_2 überdecken. Sei $\{T_{i,j}, V_{s,t}\} \in E_1$. Falls $V_{s,t} \in S$, dann wird diese Kante überdeckt. Sonst entspricht nach der Konstruktion von S der Knoten $V_{s,t}$ einem von α auf 1 gesetzten Literal, also ist $T_{i,j} \in S$. Die Argumentation für Kanten aus E_2 ist analog.

Sei nun S ein Vertex-Cover für G_ϕ der Grösse $5q$. Es sind mindestens $2q$ Knoten nötig, um alle Klausel-Dreiecke Δ_s zu überdecken, und mindestens $3q$ Knoten, um alle Kreise H_i zu überdecken. Also enthält S in jedem Klausel-Dreieck Δ_s genau zwei Knoten und (gemäss

der Beobachtung oben) in jedem Kreis H_i entweder alle Knoten $T_{i,1}, T_{i,2}, \ldots, T_{i,m(i)}$ oder alle Knoten $F_{i,1}, F_{i,2}, \ldots, F_{i,m(i)}$. Wir definieren nun eine Belegung α durch $\alpha(x_i) = 1$, falls $T_{i,1} \in S$, und $\alpha(x_i) = 0$, falls $F_{i,1} \in S$.

Wir müssen nun noch zeigen, dass α die Formel ϕ erfüllt. Angenommen, es gibt eine Klausel C_s in ϕ, die von α nicht erfüllt wird. Zur Vereinfachung der Argumentation gehen wir davon aus, dass C_s nur positive Literale enthält. In allen anderen Fällen lässt sich analog argumentieren. Sei also $C_s = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$, wobei dies das j_1-te, j_2-te und j_3-te Vorkommen der jeweiligen Variablen sei. Weil C_s nicht erfüllt ist, gilt $\alpha(x_{i_1}) = \alpha(x_{i_2}) = \alpha(x_{i_3}) = 0$. Also sind die Knoten T_{i_1,j_1}, T_{i_2,j_2} und T_{i_3,j_3} nach Konstruktion nicht in S enthalten. Die drei Kanten $\{T_{i_1,j_1}, V_{s,1}\}$, $\{T_{i_2,j_2}, V_{s,2}\}$ und $\{T_{i_3,j_3}, V_{s,3}\}$ müssen also durch die drei Knoten $V_{s,1}$, $V_{s,2}$ und $V_{s,3}$ überdeckt werden, von denen aber nach Konstruktion nur zwei in S enthalten sind. Dies ist ein Widerspruch dazu, dass S ein Vertex-Cover ist. Also war unsere Annahme falsch und C_s wird von α erfüllt. Damit ist α eine erfüllende Belegung für ϕ.

Lösung zu Aufgabe 32

Sei $L \in \mathrm{VP}$, sei A ein Polynomialzeit-Verifizierer für L. Es gelte, dass für jedes Wort $w \in L$ ein Zeuge x mit $|x| \leq \log_2 |w|$ existiert, so dass A die Eingabe (w, x) akzeptiert.

Die folgende Turingmaschine M entscheidet L: Für alle Zeugen $x \in \Sigma^*_{\mathrm{bool}}$ mit $|x| \leq \log_2 |w|$ simuliert M nacheinander die Arbeit von A auf (w, x). Falls A ein Paar (w, x) akzeptiert, dann akzeptiert auch M. Falls A jedes Paar (w, x) verwirft, dann verwirft auch M.

Wir zeigen zunächst $L(M) = L$. Wenn $w \in L$, dann existiert nach Voraussetzung ein Zeuge $x \in \Sigma^*_{\mathrm{bool}}$ mit $|x| \leq \log_2 |w|$, so dass (w, x) von A akzeptiert wird. Dieser Zeuge wird auch von M in einer ihrer Simulationen untersucht, also akzeptiert M in dieser Simulation. Damit gilt $w \in L(M)$.

Wenn $w \in L(M)$, dann hat M in einer ihrer Simulationen ihre Eingabe akzeptiert. Dies geschieht aber nur dann, wenn für den in dieser Simulation verwendeten Zeugen x der Verifizierer A die Eingabe (w, x) akzeptiert hat. Da nur Zeugen mit $|x| \leq \log_2 |w|$ verwendet werden, ist damit die Bedingung für $w \in L$ erfüllt.

Nun zeigen wir, dass M in Polynomzeit läuft. Da A ein Polynomialzeit-Verifizierer ist, existiert ein Polynom p mit $\mathrm{Time}_A(w, x) \leq p(|w|)$ für alle $x \in \Sigma^*_{\mathrm{bool}}$. Die Laufzeit einer Simulation von M ist also beschränkt durch $p(|w|) + c$ für eine Konstante c.

Es ergibt sich als Anzahl der Zeugen $x \in \Sigma^*_{\mathrm{bool}}$ mit $|x| \leq \log_2 |w|$:

$$\sum_{i=0}^{\lfloor \log_2 |w| \rfloor} 2^i = 2^{\lfloor \log_2 |w| \rfloor + 1} - 1 < 2^{\lfloor \log_2 |w| \rfloor + 1} \leq 2|w|.$$

Die Anzahl der Zeugen $x \in \Sigma^*_{\mathrm{bool}}$ mit $|x| \leq \log_2 |w|$ ist aber gerade die maximale Anzahl der Simulationen, die M durchführt, also folgt insgesamt für zwei Konstanten c und d, dass

$$\mathrm{Time}_M(w) \leq 2|w| \cdot (p(|w|) + c) + d \in O(|w| p(|w|)).$$

Damit ist M eine Polynomialzeit-TM, die L entscheidet, also $L \in \mathrm{P}$.

Lösung zu Aufgabe 33

Wir verwenden für jedes Feld (i, j) des $(n \times n)$-Schachbretts eine Variable $x_{i,j}$. Wenn $x_{i,j}$ mit 1 belegt wird, dann interpretieren wir das als die Platzierung eines Turms auf das Feld (i, j). Damit entspricht jede Belegung der n^2 Variablen einer möglichen Platzierung von bis zu n^2 Türmen auf dem Schachbrett.

Damit die platzierten Türme sich nicht gegenseitig bedrohen, darf in jeder Zeile und in jeder Spalte höchstens ein Turm stehen.
Dies lässt sich durch die folgenden Formeln beschreiben:

- Für jede Zeile i verwenden wir die Teilformel

  $$Z_i = \bigwedge_{1 \leq j_1 < j_2 \leq n} (\overline{x_{i,j_1}} \vee \overline{x_{i,j_2}}).$$

  Dadurch ist sichergestellt, dass keine zwei Variablen der Zeile i gleichzeitig mit 1 belegt werden können.

- Analog verwenden wir für jede Spalte j die Teilformel

  $$S_j = \bigwedge_{1 \leq i_1 < i_2 \leq n} (\overline{x_{i_1,j}} \vee \overline{x_{i_2,j}}).$$

Die Konjunktion

$$F_1 = \bigwedge_{1 \leq i \leq n} Z_i \wedge \bigwedge_{1 \leq j \leq n} S_j$$

ist dann eine KNF-Formel und beschreibt die Bedingung, dass sich je zwei platzierte Türme nicht bedrohen.
Um zusätzlich zu garantieren, dass genau n Türme platziert wurden, beobachten wir, dass dies nur dann der Fall sein kann, wenn jede Zeile genau einen solchen enthält. Weil die oben beschriebene Formel schon garantiert, dass höchstens ein Turm pro Zeile platziert wird, reicht es aus, noch eine Formel hinzuzufügen, die dafür sorgt, dass pro Zeile mindestens ein Turm platziert wird. Dies lässt sich für Zeile i durch die Formel

$$\bigvee_{1 \leq j \leq n} x_{i,j}$$

beschreiben.
Damit ergibt sich insgesamt die gesuchte Formel

$$F = F_1 \wedge \bigwedge_{1 \leq i \leq n} \bigvee_{1 \leq j \leq n} x_{i,j}.$$