Uebungsblatt 06 - Theoretische Informatik 2

Kügler, Lennart - Informatik - 5236372
l.kuegler@tu-braunschweig.de

Rausch, Moritz - Informatik - 5155947
moritz.rausch@tu-braunschweig.de

Schäfer, Maximilian - Elektrotechnik - 4763431
maximilian.schaefer@tu-braunschweig.de

Viradia, Yash - Informatik - 5275038
y.viradia@tu-braunschweig.de

10.07.2023

7)

a)

<u>Vereinigung</u>: L_1 oder L_2 muss akzeptieren

Seien $L_1, L_2 \in P$, M_1, M_2 DTM mit $L(M_1) = L_1$ und $L(M_2) = L_2$

Sei M eine DTM, die M_1 und M_2 als Eingabe erhält:

- Kopiere Eingabe auf Band 2
- Simuliere M_1 auf Band 1
  - Wenn M_1 akzeptiert, dann akzeptiere
- Wenn M_1 ablehnt, dann simuliere M_2 auf Band 2
  - Wenn M_2 akzeptiert, dann akzeptiere
  - Wenn M_2 ablehnt, dann lehne ab

$$L(M) = L(M_1) \cup L(M_2)$$

Da L_1 und $L_2 \in P$ sind lassen sich beide in $O(n^k)$ lösen
Beide zusammen brauchen also $O(n^{k_1}) + O(n^{k_2})$ und $O(n)$ für das Kopieren auf Band 2, also insgesamt $O(n^{\max\{k_1, k_2\}})$.

Die Bandanzahl verändert das nicht, da P eine robuste Klasse ist und daher gegenüber exponentiellem Blowup abgeschlossen ist

Konkatenation:

Seien $L_1, L_2 \in P$, M_1, M_2 DTM mit $L(M_1) = L_1$ und $L(M_2) = L_2$

$w = w_1 \cdot w_2$

Überprüfe, ob es eine Aufteilung von w gibt, sodass $w_1 \in L_1$ und $w_2 \in L_2$

Falls $\varepsilon \in L_1$ und $\varepsilon \in L_2$, gibt es $|w|+1$ verschiedene Aufteilungen, also $O(n)$ mit $n = |w|$

Vorgehen in DTM M:

- Kopiere die Eingabe auf $n+1$ verschiedene Bänder $\} O(n) \cdot O(n) = O(n^2)$
- Simuliere auf jedem Band M_1 auf w_1, M_2 auf w_2 $\} (O(n^{k_1}) + O(n^{k_2})) \cdot O(n)$
  $= O(n^{\max\{k_1,k_2\}}) \cdot O(n)$
  $= O(n^{\max\{k_1,k_2\}+1})$
  - erhöhe die Trennstelle zwischen w_1 und w_2 auf jedem Band um 1
    - speichere die Trennstelle auf einem eigenen Band

$\Downarrow$

$\in O(n^\ell), \ell \in \mathbb{N}$

$\Rightarrow L_1 \cdot L_2 \in P$

- Akzeptiere, wenn M_1 und M_2 auf einem Band akzeptieren, sonst lehne ab

Die Bandanzahl verändert das nicht, da P eine robuste Klasse ist und daher gegenüber exponentiellem Blowup abgeschlossen ist.

Komplement

Sei $L_1 \in P$, M_1 DTM mit $L(M_1) = L_1$

Erstelle M_1', indem akzeptierende und ablehnende Zustände in M_1 vertauscht werden. $L(M_1') = \overline{L_1}$

M_1' hat die gleiche Laufzeit wie M_1, also ist $\overline{L_1} \in P$

ome

b) $L \in P \Rightarrow L^* \in P$

Idee: Wir betrachten ein Wort $w = w_0 \cdot w_1 \cdots w_k$ als einen Graphen.

z.B.

Für je zwei Knoten $\{w_i, w_j \mid i<k\}$ führen wir eine Kante $w_i \longrightarrow w_j$, falls das Teilwort $w_{i+1} \cdots w_j$ in L ist.

Wir benutzen w_0 als einen besonderen Knoten, der für das leere Wort steht ($w_0 = \varepsilon$, $w_1 \cdots w_k \in \Sigma$).

Beh: $w \in L^*$ gdw. es gibt einen Pfad von w_0 nach w_k in Graphen.

Bew:

„$\Rightarrow$" Sei $w \in L^*$, dann Zerlegung $w = v_1 \cdot v_2 \cdots v_m$ mit $v_i \in L$, oder $w \in \varepsilon$. Betrachte $i \in \{1, \ldots m\}$

Fall 1: $w \in \varepsilon$: Der dazugehörige Graph ist w_0.
Trivialerweise w_0 von w_0 aus erreichbar

Fall 2: ($w = v_1 \cdot v_2 \cdots v_m$, $v_i \neq \varepsilon$, $v_i \in L$, für $i \in \{1, \ldots, m\}$)

Die einzelnen v_i's haben die Form $v_i = w_{n_i} \cdot w_{n_{i+1}} \cdots w_{l_i}$ mit $n_1 = 1$

$l_i + 1 = n_{i+1}$ $(i<m)$

$l_m = k$

Da $v_{i+1} \in L$ ist, $w_{n+1} \cdots w_{l_i+1} \in L$, also $w_{l_i+1} \cdots w_{l_i+1} \in L$

und nach Konstruktion des Graphen gibt es Kante (w_{l_i}, w_{l_i+1})

~~$(\forall i : 1 \leq i \leq m)$~~ Betrachte $1 \leq i < m$.

Außerdem $v_i \in L$ $v_1 = w_1 \cdots w_{l_1}$. Dann gibt es die Kante (w_0, w_{l_1})

Damit ist $w_0 \to w_{l_1} \to \cdots \to w_{l_m} = w_k$ ein Pfad im Graphen von w_0 nach w_k.

„$\Leftarrow$" Wenn es ein Pfad $w_0 \to^{\infty} w_k$ gibt, dann können wir entlang ~~benutzten~~ der benutzten Kanten zerlegen und erhalten eine Zerlegung $w = v_1 \cdots v_m$ mit $v_i \in L$ für $i \in \{1, \cdots, m\}$

Nun lässt sich DTM M bauen, die L^* in Polyzeit erkennt.

Konstruiere Graph G

$O(n)$ Knoten
$O(n^2)$ Kanten, je Kante in $time_{M_L}(n)$ berechnen.

Führe PATH auf G aus. Benötigt Polyzeit

$$time_M(n) \approx \underbrace{O(n^2) \cdot time_{M_L}(n)}_{\text{Konstruktion}} + \underbrace{poly(n)}_{\text{PATH}}$$

PATH $\in$ NL, NL $\subseteq$ P, PATH in P lösbar $\lightning$

2)

a)

Vertex-Covering kann mit einem Polynomialzeit-Verifizierer gelöst werden.

Dazu wird als Zertifikat eine Menge $S \subseteq V$ mit $|S| = k$ gerateten.
Der Verifizierer prüft, ob für jede $e \in E$, $e = \{v, w\}$ mit $v, w \in V$
v oder w in S liegt.

Das überprüfen ist zeitlich polynomiell beschränkt, da für jede Kante
S zwei mal durchlaufen werden muss und $|S| \leq |V|$ ist.

Da VC sich durch einen Polynomialzeit-Verifizierer lösen lässt,
liegt es laut Theorem 11.8 in NP

b)

Wie in der glü gezeigt ist $SAT \leq_m^{log} 3SAT$. D.h wenn $3SAT \leq_m^{log} VC$ gilt auch $SAT \leq_m^{log} VC$

Reduktion von 3SAT auf VC:

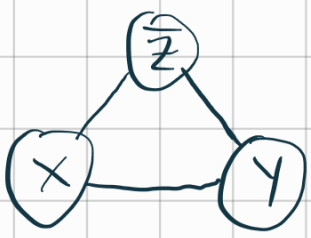
Eingabe: eine Formel φ in CNF m/A 3 Literalen pro Klausel

- Für jede Variable x erstelle je einen Knoten für x und $\bar{x}$ und verbinde die beiden.

  zB. für die Variablen x und y:

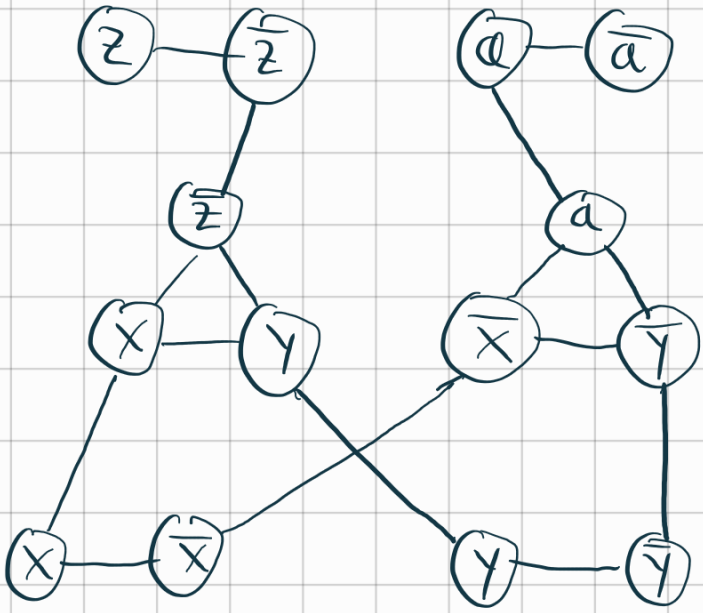
- Für jede Klausel erstelle einen Knoten für jedes der 3 Literale und verbinde diese 3 miteinander

  z.B. für die Klausel $(x \vee y \vee \bar{z})$:



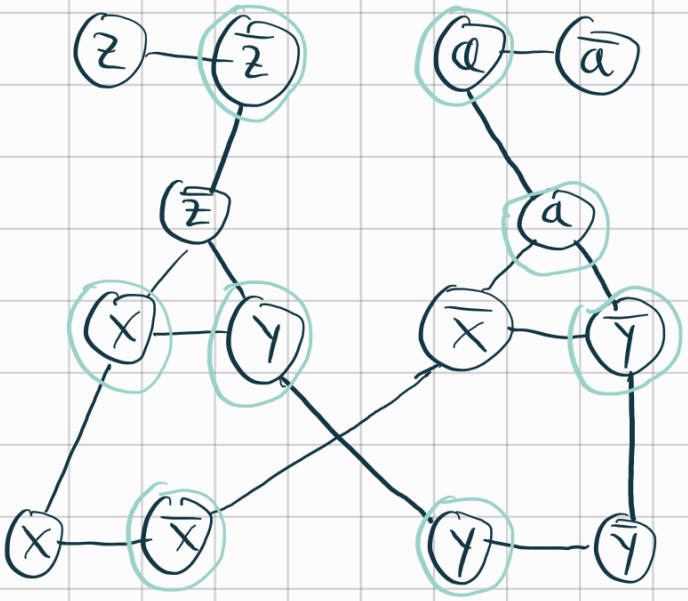
- Verbinde jedes Literal in den Variablen-Graphen mit deren Vorkommen in den Klausel-Graphen

z.B. für $\varphi = (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee a)$



Setze $k = 2c + v$ mit c = Anzahl der Klauseln,
v = Anzahl der Variablen

z.B. $k = 2c + v = 2 \cdot 2 + 4 = 8$



- Wähle aus den Variabel-Tupeln jeweils einen Knoten.
  $\bar{x}$ ausgewählt entspricht $\bar{x} = T \Leftrightarrow x = \bot$

- Wähle aus den Klausel-Tripeln jeweils zwei Knoten. Der nicht ausgewählte Knoten muss mit einem ausgewählten Knoten aus einem Variablen-Graph verbunden sein. Solch ein Knoten existiert auf jeden Fall*, da pro Klausel mindestens ein Literal wahr sein muss.

* für eine erfüllbare SAT-Instanz

c)

(1) VALIDITY $\in$ coNP $\Longleftrightarrow$ $\overline{\text{VALIDITY}} \in$ NP

$\overline{\text{VALIDITY}}$ kann mit einem Polynomialzeit-Verifizierer gelöst werden:

- Erstelle Zertifikat:
  rate eine Belegung der Variablen von φ
- Setze die Belegung ein und überprüfe, ob φ wahr ist

Dies kann nichtdeterministisch für jede mögliche Belegung ausgeführt werden. Ist φ für eine der Belegungen falsch, akzeptiert die NTM, ist φ für alle Belegungen wahr, lehnt sie ab.

Laut Theorem 11.8 liegt $\overline{\text{VALIDITY}}$ dadurch in NP
und VALIDITY somit in coNP

(2)

φ ist erfüllbar $\Longleftrightarrow$ $\neg\varphi$ ist keine Tautologie

Also gibt es eine Reduktion f von SAT auf VALIDITY:

$$f(\varphi) = \neg\varphi$$

$$\Rightarrow \quad \text{SAT} \leq_m^{\log} \overline{\text{VALIDITY}}$$

Da SAT NP-vollständig ist, gilt für alle $L \in$ NP

$L \leq_m^{log} SAT \leq_m^{log} \overline{VALIDITY}$

$\Rightarrow L \leq_m^{log} \overline{VALIDITY}$ mit $\exists$ Reduktion g

also

$$x \in L \iff g(x) \in \overline{VALIDITY}$$

gdw. $$x \notin L \iff g(x) \notin \overline{VALIDITY}$$

gdw. $$x \in \bar{L} \iff g(x) \in VALIDITY$$

Da $L \in NP$ ist $\bar{L} \in coNP$.

Also gibt es für alle $\bar{L} \in coNP$ eine Reduktion g auf VALIDITY und somit ist VALIDITY coNP-schwer bzgl. $\leq_m^{log}$.

$\Rightarrow$ VALIDITY ist coNP-vollständig bzgl. $\leq_m^{log}$