Theoretische Informatik 2

Tutor: Martin Klossek

6. Übungsblatt

Adam Duck

13. Juni 2002

6.1 2-SAT Problem

Eine 2-`KNF` ist eine Formel F in konjunktiver Normalform mit der Eigenschaft, dass jede Klausel genau zwei Literale besitzt. Sei

$$2 - \mathsf{SAT} = \{F \mid F \text{ ist eine 2-}\texttt{KNF} \text{ und } F \text{ ist erfüllbar.}\}$$

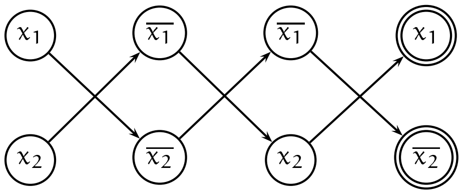
Zeige, dass 2-SAT in $\mathcal{P}$ liegt.
Lösung:
Eine 2-`KNF` hat die folgende Form:

z_1 $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$.
Hier wäre $V = \{x_1, x_2, \overline{x_1}, \overline{x_2}\}$ und eine mögliche Lösung $\binom{x_1}{x_2} = \binom{1}{0}$ oder auch $\binom{0}{1}$.

z_2 $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2})$.
z_2 ist nicht erfüllbar, denn mit jeder der vier Möglichkeiten können wir höchstens drei Klauseln erfüllen.

Nun haben wir zwei Mögliche Lösungen.

- Dieser Algorithmus baut einen Graphen von links nach rechts folgendermaßen auf (hier zu z_2):



  Normalerweise würde der Algorithmus alle Verbindungen von der k-ten zur $(k+1)$-ten Klausel einfügen, jedoch sind Kanten der Form $x_1 \longrightarrow \overline{x_1}$ unsinnig. Nun sucht man mittels Tiefensuche einen Weg von der ersten Klausel zur letzten. Gelingt das, und ist der so entstandene Ausdruck ohne Widersprüche, so ist dieses 2-SAT lösbar.

- Die zweite Lösung macht von der folgenden Tatsache Gebrauch:

$$\begin{aligned} x \vee y &\equiv \neg x \Rightarrow y \text{ bzw.} \\ y \vee x &\equiv \neg y \Rightarrow x \end{aligned}$$

6.2 Abschlusseigenschaften von $\mathcal{P}$
Seien L_1 und L_2 in $\mathcal{P}$. **Welche** der Sprachen $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \circ L_2$, $\overline{L_1}$, L_1^* sind dann auch in $\mathcal{P}$?
Lösung:

$\overline{L_1}$ Die `TM` gibt entweder `TRUE` oder `FALSE` aus, da sie ja mit ihrem Programm überprüft, ob ein bestimmtes Wort w in L_1 ist oder nicht. Wir müssen also nur diesen Output umkehren und haben eine `TM`, die $\overline{L_1}$ erkennt.

$L_1 \cup L_2$ Wir bauen eine `TM`, die nur dann ein Wort erkennt, wenn beide Touringmaschinen $\texttt{TM}_1$ – die Wörter aus L_1 erkennt – und $\texttt{TM}_2$ – die Wörter aus L_2 erkennt – das Wort erkennen.

$L_1 \cap L_2$ Hier erkennen wir alle Wörter, die von $\texttt{TM}_1$ oder von $\texttt{TM}_2$ erkannt werden.

$L_1 \circ L_2$ Hier wird die Sache schon etwas komplizierter. Wir zerlegen das Wort w in $w_1 w_2 \dots w_n$ Somit bekommen wir eine Menge

$$E = \{\varepsilon | w_1 w_2 \dots w_n, w_1 | w_2 w_3 \dots w_n, \dots, w_1 w_2 \dots w_{n-1} | w_n, w_1 w_2 w_3 \dots w_n | \varepsilon\}.$$

Nun überürüfen wir, ob $\text{Präfix}_k(w) \in L_1 \wedge \text{Suffix}_k(w) \in L_2$. Da $|E| = |w|+1$ sind wir immernoch polynomiell.

L_1^* Hier zerlegen wir w in wiederum in $w_1 w_2 \dots w_n$ und bilden die Potenzmenge

$$W = \{w_i \dots w_j \mid 1 \leq i \leq j \leq n\}$$

Desweiteren prüfen wir $\forall v \in W$, ob $v \in L_1$. Wenn ja, setzen wir $X := X \cup \{(i, j)\}$.

Nun können wir

- $G = (V, E)$
- $V = \{0, 1, \dots, n\}$
- $E = \{(i, j) \mid j > i \wedge (i + 1, j) \in X\}$

bilden und mittels Tiefensuche vpm Anfang bis zum Ende einen Weg suchen. Ist so einer gefunden, ist das Wort in L_1^*.

Da die Potenzmengenkonstruktion $\mathcal{O}(n^2)$ benötigt, ist die Gesamtlaufzeit $\mathcal{O}(n^2) \cdot \mathtt{poly}(n)$.

6.3 Entscheiden vs. Optimieren
Wir betrachten in dieser Aufgabe drei verschiedene Varianten des Problems SET PACKING: Sei $\mathcal{A} = \{A_1, \dots, A_m\}$ eine Menge endlicher Mengen. Die Länge der Eingabe ist $\sum_{i=1}^{m} |A_i|$. Untersucht werden disjunkte Teilmengen aus $\mathcal{A}$.

Variante 1: Enthält $\mathcal{A}$ mindestens k disjunkte Teilmengen? Hierbei ist die natürliche Zahl k Teil der Eingabe.

Variante 2: Berechne die maximale Anzahl disjunkter Teilmengen aus $\mathcal{A}$.

Variante 3: Gib eine größtmögliche Menge von disjunkten Teilmengen aus $\mathcal{A}$ aus.

Die erste Variante heißt auch Entscheidungsvariante. Die zweite Variante fragt nach dem Wert einer optimalen Lösung. Die dritte Variante erfordert die Berechnung einer optimalen Lösung (das Suchproblem).

(a) Angenommen Variante 1 ist in $\mathcal{P}$. Ist dann auch Variante 2 deterministisch in Polynomialzeit lösbar? Begründe deine Antwort.
Lösung:
Ja, da man für jedes $k = 0, \ldots, n$ immer "Variante 1" aufrufen kann. Dann nimmt man das höchste k, für das "Variante 1" `TRUE` zurückgegeben hat.

(b) Angenommen Variante 1 ist in $\mathcal{P}$. Ist dann auch Variante 3 deterministisch in Polynomialzeit lösbar? Begründe deine Antwort.
Lösung:
Auch hier ist eine Lösung in polynomieller Zeit machbar:

`INIT`: $\mathsf{OPT} := \emptyset$, $\mathcal{A}_i = \mathcal{A} \setminus A_i$

`TEST`: $\forall A_i \in \mathcal{A}_i$:

wenn $k_{max}(\mathcal{A}) \neq k_{max}(\mathcal{A}_i)$

dann $\mathsf{OPT} := \mathsf{OPT} \cup \{A_i\}$

sonst $\mathcal{A} := \mathcal{A} \setminus A_i$

6.4 $\mathcal{NP}$ als eine Projektion von $\mathcal{P}$

Sei $R \subseteq A^* \star B^*$ eine Relation. Die Relation R ist *polynomiell beschränkt*, falls es ein Polynom $p(\cdot)$ gibt, so dass $\forall (x, y) \in R$ gilt $|y| \leq p(|x|)$. Die zugehörige Sprache $L(R)$ ist wie folgt definiert

$$L(R) := \{x \mid \exists y (x, y) \in R\}$$

($x \in L(R)$ gilt also genau dann, wenn es einen *Zeugen* y mit $(x, y) \in R$ gibt.)
Wir sagen, dass R in $\mathcal{P}$ liegt, falls die Zugehörigkeit eines Paares (x, y) zu R in *polynomieller* Zeit entschieden werden kann.
Zeige, dass

$$\mathcal{NP} = \{L(R) \mid R \in \mathcal{P} \text{ und } R \text{ ist polynomiell beschränkt}\}.$$

Lösung:
Das wollte nicht einmal der Tutor vorführen $\smile$.