Theoretische Informatik
4. Übungsblatt

Paul Winkler
11818749

Aufgabe 1. Wir zeigen im Folgenden, dass 2SAT $\in \mathbf{P}$. Zuerst beobachten wir, dass wir o. B. d. A. davon ausgehen können, dass jede Klausel zwei verschiedene Literale enthält. Tritt nämlich z. B. p_1 vereinzelt auf, müssen wir es, damit $\mathcal{C}$ erfüllbar sein kann, in der gesamten Formel auf »wahr« setzen. Das können wir solange durchführen, bis wir entweder einen Widerspruch erhalten oder kein Literal mehr vereinzelt auftritt. Da wir nur endlich viele Variablen haben, benötigt diese Vorbereitung polynomielle Zeit (in der Anzahl der Variablen).

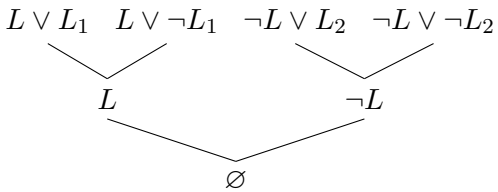
Wir bauen nun für eine gegebene Klauselmenge $\mathcal{C}$ den folgenden gerichteten Graphen $G_{\mathcal{C}}$: Die Knoten sind $\textcircled{p}$ und $\textcircled{\neg p}$ für jede in $\mathcal{C}$ vorkommende Variable p. Für jede in $\mathcal{C}$ enthaltene Klausel $\{L_1, L_2\}$ inkludieren wir die Kanten $\neg L_1 \to L_2$ und $\neg L_2 \to L_1$. Wir zeigen nun:

$\mathcal{C}$ erfüllbar $\iff$ Es gibt kein Literal L, sodass $G_{\mathcal{C}}$ einen gerichteten Pfad von L nach $\neg L$ enthält.

»$\Rightarrow$« $\mathcal{C}$ wird von einer Belegung b genau dann erfüllt, wenn jede Klausel mindestens ein unter b wahres Literal enthält. Eine Kante $L \to L'$ steht für die Klausel $\{\neg L, L'\}$; es gilt also: Sei b eine Belegung, die $\mathcal{C}$ erfüllt, dann gilt $b(L) = 1 \implies b(L') = 1$. Ein Pfad von L nach $\neg L$ würde also $b(L) = 1 \implies b(\neg L) = 1$ implizieren, ein Widerspruch.

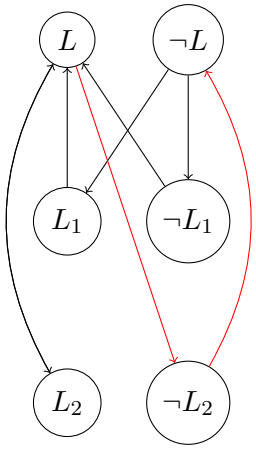
»$\Leftarrow$« Mit Kontraposition: Sei $\mathcal{C}$ unerfüllbar, dann gibt es eine Resolutionsableitung von $\varnothing$, wobei $\varnothing = \mathrm{Res}(\{L\}, \{\neg L\})$ sei. Wir können uns die Resolutionsableitung als auf dem Kopf stehenden Ableitungsbaum vorstellen, der die Klauseln von $\mathcal{C}$ als Blätter hat. Dabei hat jeder innere Knoten entweder zwei Eltern (falls die Klausel die Resolvente von zwei Klauseln ist) oder nur sich selbst als Vater. Die $(i+2)$-te Ebene dieses Baumes (von unten) können wir als Klauselmenge $\mathcal{C}_i$ interpretieren. Wir zeigen nun induktiv für alle n Ebenen, dass $G_{\mathcal{C}_i}$ einen gerichteten Pfad von L nach $\neg L$ enthält, insbesondere enthält für $i = n$ also auch $G_{\mathcal{C}_n} = G_{\mathcal{C}}$ einen solchen Pfad.

- $i = 1$: Der unterste Teil des Ableitungsbaumes hat die Form



und der zugehörige Graph ist

$G_{\mathcal{C}_1}$:



Er enthält den gerichteten Pfad $L \to \neg L_2 \to \neg L$ (rot eingezeichnet).

- $i \to i+1$: Für den Induktionsschritt reicht es zu zeigen, dass es für jede Kante $L_m \to L_k$ in $G_{\mathcal{C}_i}$ einen gerichteten Pfad von L_m nach L_k in $G_{\mathcal{C}_{i+1}}$ gibt. Dass $L_m \to L_k$ eine Kante in $G_{\mathcal{C}_i}$ ist, ist gleichbedeutend damit, dass der Ableitungsbaum einen der beiden folgenden Teilbäume besitzt:

$$\begin{array}{cc} L_p \vee \neg L_m \quad \neg L_p \vee L_k & \neg L_m \vee L_k \\ \diagdown\!\diagup & | \\ \neg L_m \vee L_k & \neg L_m \vee L_k \end{array}$$

Im linken Fall enthält $G_{\mathcal{C}_{i+1}}$ den gerichteten Pfad $L_m \to L_p \to L_k$, im rechten Fall selbst die Kante $L_m \to L_k$.

Wir können also aus unserer ursprünglichen Klauselmenge $\mathcal{C}$ den Implikationsgraphen $G_{\mathcal{C}}$ aufbauen und dann für jedes Literal überprüfen, ob es einen gerichteten Pfad von diesem zum jeweiligen negierten Literal gibt. Das Aufbauen des Graphen benötigt klarerweise polynomielle Zeit (in der Anzahl der Klauseln), das Durchsuchen nach den Pfaden ebenfalls (wir können z. B. Breitensuche verwenden).

Aufgabe 2. Sei $\mathcal{C}$ eine Klauselmenge, $C \in \mathcal{C}$ eine Klausel, $L_1, L_2 \in C$ und $C_0 := C \backslash \{L_1, L_2\}$. Sei p eine neue aussagenlogische Variable, dann wollen wir $L_1 \vee L_2$ durch p simulieren, d. h. durch p ersetzen und zusätzlich die Bedingung $p \leftrightarrow L_1 \vee L_2$ einbauen. Wir erhalten

$$p \to L_1 \vee L_2 \iff \underbrace{\neg p \vee L_1 \vee L_2}_{=: C_1},$$

$$L_1 \vee L_2 \to p \iff \neg(L_1 \vee L_2) \vee p \iff (\neg L_1 \wedge \neg L_2) \vee p \iff \underbrace{(\neg L_1 \vee p)}_{=: C_2} \wedge \underbrace{(\neg L_2 \vee p)}_{=: C_3}.$$

$\mathcal{C}$ ist nun erfüllbar genau dann, wenn $\mathcal{C} \backslash \{C\} \cup \{C_0 \cup p, C_1, C_2, C_3\}$ erfüllbar ist:

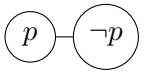
- Für die eine Implikation sei b eine Belegung, die $\mathcal{C}$ erfüllt. Dann definieren wir eine neue Belegung $\tilde{b} := b \cup \{p \to b(L_1) \vee b(L_2)\}$. Nach Voraussetzung werden die Klauseln in $C \backslash \{C\}$ auch von $\tilde{b}$ erfüllt und nach Definition von $\tilde{b}$ auch die übrigen Klauseln.

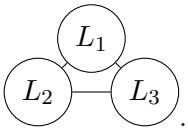
- Für die andere Richtung sei b eine Belegung, die $\mathcal{C} \backslash \{C\} \cup \{C_0 \cup p, C_1, C_2, C_3\}$ erfüllt. Weil insbesondere C_1, C_2 und C_3 von b erfüllt werden, gilt $b(p) = b(L_1) \vee b(L_2)$. Wir können also $C_0 \cup p$ durch $C_0 \cup \{L_1, L_2\} = C$ ersetzen und sehen, dass b auch $\mathcal{C}$ erfüllt.

Daraus folgt nun, dass 3SAT **NP**-vollständig ist, denn mit der obigen Konstruktion können wir SAT in polynomieller Zeit auf 3SAT zurückführen: Sei dazu C eine Klausel mit $n > 3$ Literalen, dann führen wir $n - 3$ neue aussagenlogische Variablen ein und ebensooft die oben angegebene Ersetzung durch. In jedem Schritt verringert sich die Anzahl der Literale in der ursprünglichen Klausel um eins und es werden nur Klauseln mit höchstens drei Literalen neu hinzugefügt. Jede der $n - 3$ Ersetzungen hat konstanten Aufwand. Damit ist der Aufwand für jede Klausel polynomiell in der Anzahl der Literale und damit auch für jede Klauselmenge polynomiell in der Gesamtanzahl der Literale.

Wieso ist Knotenüb. in NP?

Aufgabe 3. Das Knotenüberdeckungsproblem ist die Frage nach der Existenz einer Knotenüberdeckung mit $\leqslant k$ Knoten für einen endlichen ungerichteten Graphen G und eine natürliche Zahl k. Wir zeigen nun, dass es **NP**-vollständig ist, indem wir 3SAT darauf zurückführen. Sei dazu $\mathcal{C}$ eine Klauselmenge, in der jede Klausel mindestens drei Literale enthält. Wir konstruieren daraus folgenden Graphen $G_{\mathcal{C}}$, der aus einer linken und einer rechten Knotenhälfte besteht:

Für jede vorkommende Variable p verwenden wir auf der linken Seite den Baustein 

und für jedes Literal $L_1 \vee L_2 \vee L_3$ auf der rechten Seite den Baustein 

.

jede Klausel

Sollte eine Klausel weniger als drei Literale enthalten, verwenden wir trotzdem den Baustein mit drei Knoten und beschriften eben zwei oder drei Knoten mit demselben Literal. Zusätzlich verbinden wir jeden Knoten auf der rechten Seite mit dem Knoten der entsprechenden Variablen oder negierten Variablen auf der linken Seite. Dass wir aus einer Klauselmenge den entsprechenden Graphen in polynomieller Zeit aufbauen können, ist offensichtlich.

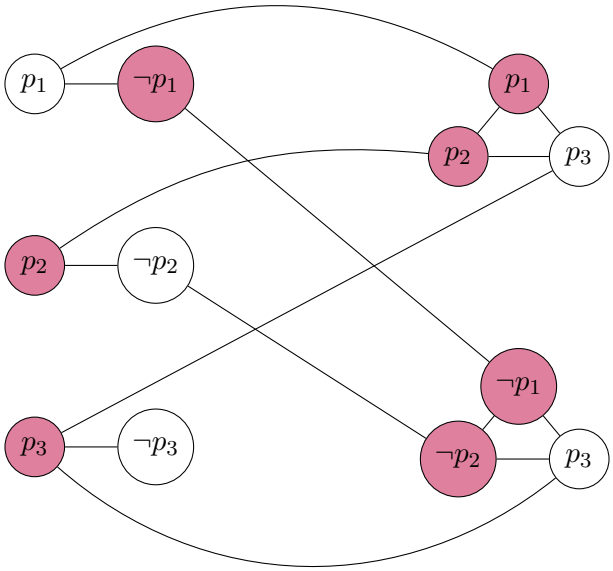
Eine beliebige Knotenüberdeckung dieses Graphen muss auf der linken Seite für jede Variable p entweder »p« oder »$\neg p$« und auf der rechten Seite für jede Klausel mindestens zwei der drei entsprechenden Literale enthalten. Sei n die Anzahl der Variablen und m die Anzahl der Klauseln in $\mathcal{C}$, dann besteht jede Knotenüberdeckung also aus mindestens $n + 2m$ Knoten. Es gilt nun:

$\mathcal{C}$ erfüllbar $\iff$ es gibt eine Knotenüberdeckung von $G_{\mathcal{C}}$ aus genau $n + 2m$ Knoten.

»$\Rightarrow$« Sei $\mathcal{C}$ erfüllbar, dann wählen wir folgende Knotenüberdeckung: Auf der linken Seite wählen wir genau die unter der einschlägigen Belegung wahren Literale aus. Auf der rechten Seite hat jede Klausel mindestens ein Literal, das unter der entsprechenden Belegung auf »wahr« gesetzt wird; wir wählen also für jede Klausel genau ein so ein Literal aus, dass wir *nicht* in die Knotenüberdeckung aufnehmen. Ok. Wieso ist das eine Knotenüberdeckung?

»$\Leftarrow$« Sei V' eine Knotenüberdeckung aus $n + 2m$ Knoten. Weil das, wie wir vorher gezeigt haben, die minimal mögliche Anzahl an Knoten ist, müssen auf der linken Seite für jedes p genau »p« oder »$\neg p$« und für jede Klausel auf der rechten Seite genau zwei der drei Literale in V' liegen. Wir wählen nun folgende Belegung: Wird auf der linken Seite der Knoten »p« ausgewählt, so wird p auf »wahr« gesetzt, wird hingegen »$\neg p$« ausgewählt, wird p auf »falsch« gesetzt. Auf der rechten Seite werden folglich alle Knoten, die von V' *nicht* ausgewählt wurden, mit »wahr« interpretiert, denn sie sind ja über eine Kante mit demselben Literal auf der linken Seite verbunden, dass dann in V' liegen muss und damit auf »wahr« gesetzt ist – also enthält jede Klausel ein unter dieser Belegung wahres Literal.

Wir illustrieren das ganze noch an einem Beispiel: Sei $\mathcal{C} = \{\{p_1, p_2, p_3\}, \{\neg p_1, \neg p_2, p_3\}\}$, dann erhalten wir folgenden Graphen $G_{\mathcal{C}}$:



Hier ist $n = 3, m = 2$ und $\mathcal{C}$ wird erfüllt durch die Belegung $\{p_1 \mapsto 0, p_2 \mapsto 1, p_3 \mapsto 1\}$.