Theoretische Informatik

4. Übung

Richard Weiss

6.6.2021

Aufgabe 1. Eine Klausel ist eine Menge von Literalen. Ein Literal ist ein Atom oder ein negiertes Atom. Seien C und D Klauseln und p ein Atom so dass $p \in C$ und $p \in D$. Dann wird $E = (C \backslash \{p\}) \cup (D \backslash \{\neg p\})$ als *Resolvente* von C und D bezeichnet. Die *Resolutionsregel* erlaubt die Ableitung von E aus C und D. Die leere Klausel $\varnothing$ entspricht, als leere Disjunktion, der Aussage „falsch“. Eine Klauselmenge $\mathcal{C}$ ist unerfüllbar genau dann wenn sich aus $\mathcal{C}$ mittels Resolution die leere Klausel ableiten lässt. Wir betrachten das folgende Entscheidungsproblem:

2SAT

Eingabe: Klauselmenge $\mathcal{C}$ in der jede Klausel höchstens 2 Literale enthält
Frage: Ist $\mathcal{C}$ erfüllbar?

Zeigen Sie dass 2SAT $\in \mathbf{P}$ ist.

Lösung. Es gibt bloß folgende Formen der Resolution:

$$\varnothing = \mathrm{Res}(\{L\}, \{\overline{L}\}), \tag{1}$$
$$\{L\} = \mathrm{Res}(\{K\}, \{\overline{K}, L\}), \tag{2.1}$$
$$\{L\} = \mathrm{Res}(\{K, L\}, \{\overline{K}, L\}), \tag{2.2}$$
$$\{L, K\} = \mathrm{Res}(\{X, L\}, \{\overline{X}, K\}); \tag{3}$$

Nachdem wir 2SAT und nicht 3SAT betrachten, gibt es keine

$$\{L, K\} = \mathrm{Res}(\{X, L, K\}, \{\overline{X}\}), \tag{3.1}$$
$$\{L, K\} = \mathrm{Res}(\{X, L, K\}, \{\overline{X}, K\}), \tag{3.2}$$
$$\{L, K\} = \mathrm{Res}(\{X, L, K\}, \{\overline{X}, K, L\}). \tag{3.3}$$
$$\tag{1}$$

Wenn wir (1) anwenden, dann sind wir fertig.
Wenn wir (2.1) anwenden müssen, dann könnte es sein, dass $\{K\}$ erst durch (2.1) oder (2.2) erzeugt werden muss. Das werden wir auch tun.

0. Das gleiche könnte ad hoc auch für $\{\overline{K}, L\}$ und (3) gelten. $\{\overline{K}, L\}$ erhält man aber (nur) aus (3), d.h.

$$\{\overline{K}, L\} = \mathrm{Res}(\{X, \overline{K}\}, \{\overline{X}, L\}).$$

Wir haben aber bereits $\{K\}$. Also können wir stattdessen auch mit (2.1) arbeiten, d.h.

$$\{X, \overline{K}\} \xrightarrow{\mathrm{Res}(\{K\}, \cdot)} \{X\} \xrightarrow{\mathrm{Res}(\cdot, \{\overline{X}, L\})} \{L\}. \tag{4}$$

1. Wenn $\{X, \overline{K}\}$ durch (3) hervorgeht, dann sei

$$\{X, \overline{K}\} = \mathrm{Res}(\{Y, X\}, \{\overline{Y}, \overline{K}\}).$$

   Wir haben aber bereits $\{K\}$. Also können wir stattdessen auch mit (2.1) arbeiten, d.h.

$$\{\overline{Y}, \overline{K}\} \xrightarrow{\mathrm{Res}(\{K\},\cdot)} \{\overline{Y}\} \xrightarrow{\mathrm{Res}(\cdot,\{Y,X\})} \{X\}.$$

   Von da aus können wir mit (4) weitermachen.

2. Wenn $\{\overline{X}, L\}$ durch (3) hervorgeht, dann sei

$$\{\overline{X}, L\} = \mathrm{Res}(\{Y, \overline{X}\}, \{\overline{Y}, L\}).$$

   Wir haben aber bereits $\{K\}$. Wegen 1. haben wir damit auch $\{X\}$. Also können wir stattdessen auch mit (2.1) arbeiten, d.h.

$$\{Y, \overline{X}\} \xrightarrow{\mathrm{Res}(\{X\},\cdot)} \{Y\} \xrightarrow{\mathrm{Res}(\cdot,\{\overline{Y},L\})} \{L\}.$$

3. Falls $\{Y, X\}$, $\{\overline{Y}, \overline{K}\}$ oder $\{Y, \overline{X}\}$, $\{\overline{Y}, L\}$ durch (3) hervorgeht, so können wir jeweils 1. oder 2. analog darauf anwenden.

Wir können also (3), in gewisser Weise, via (2.1), umgehen. An der Stelle noch zwei Bemerkungen.

- Durch (2.1) erzeugen wir ein Singleton. (2.1) arbeitet mit einem Singleton. Es kann also sein, dass man (2.1) mit einem Erzeugnis von (2.1) anwenden muss. D.h. also, wir müssen (2.1) laufend anwenden, solange es nur geht.
- Durch (2.2) erzeugen wir auch ein Singleton. (2.2) arbeitet aber nicht mit Singletons. Es kann also nicht sein, dass man (2.2) it einem Erzeugnis von (2.1) oder (2.2) anwenden muss. D.h. also, wir müssen (2.2) und (3) bloß am Anfang ausschöpfen und uns dann erst um (2.1) kümmern.

Nun zu unserem Algorithmus als Pseudocode. Sei dazu $\mathcal{C}$ eine passende Klauselmenge, d.h. für alle $C \in \mathcal{C}$ gilt $|C| \leqslant 2$. Wir verwenden die Notation $V(G)$, $E(G)$, und $w(G)$, für die Knoten- und Kanten-Menge, bzw. Gewichts-Funktion eines Graphen G.

$\mathcal{C}_{\text{sing}} := \{C \in \mathcal{C} \mid |C| = 1\}$
$V(G_0) := \{C \in \mathcal{C} \mid |C| = 2\}$
$E(G_0) := \{\{C, D\} \mid C, D \in \mathcal{C}, C \neq D, C.D \text{ resolvierbar}, \text{Res}(C, D) \notin V_0\}$
$w(G_0) : E_0 \to \bigcup \mathcal{C} : e \mapsto \{L \mid e \text{ kann entlang } L \text{ resolviert werden.}\} \neq \varnothing$
$n := 0$
while $|E(G_n)| \neq 0$ **do**
  $G_{n+1} := G_n$
  wähle $e \in E_{n+1}$
  wähle $L \in w(G_{n+1})(e)$ und $w(G_{n+1})(e) := w(G_{n+1})(e) \backslash \{L\}$
  if $|w(G_{n+1})(e)| = 0$ **then**
    $E(G_{n+1}) := E(G_{n+1}) \backslash \{e\}$
  end
  $R := \text{Res}_L(e)$
  if $|R| = 1$ **then**
    $\mathcal{C}_{\text{sing}} := \mathcal{C}_{\text{sing}} \cup \{R\}$
  else
    $V(G_{n+1}) := V(G_{n+1}) \cup \{R\}$
    $E_{\text{new}} := \{\{V, R\} \mid v \in V(G_{n+1}),\ v, r \text{ resolvierbar},\ \text{Res}(v, R) \notin V(G_{n+1})\}$
    $E(G_{n+1}) := E(G_{n+1}) \cup E_{\text{new}}$
    $w(G_{n+1}) := w(G_{n+1}) \cup \{(e', \{L' \mid e \text{ kann entlang } L' \text{ resolviert werden}\}) \mid e' \in E_{\text{new}}\}$
  end
  $n := n + 1$
end

Algorithm 1: Ausschöpfen von (2.2) und (3)

Sei n die Anzahl der Variablen in $\mathcal{C}$. Es gibt nicht mehr als

$$\binom{n}{2} = \frac{n!}{(n-2)!2!} = \frac{n(n-1)}{2}$$

Paarmengen auf L($\mathcal{C}$). Mehr Knoten können also nicht erzeugt werden. Demnach können nicht mehr als doppelt so viele Kanten erzeugt werden. In jeder Schleifen-Instanz wird eine Kante entfernt. In jeder Schleifen-Instanz wird der ganze Graph (in polynomieller Zeit) durchlaufen. Irgendwann wird es also keine Kanten mehr geben und die Schleife terminiert (in polynomieller Zeit).

$\mathcal{C}_{\text{pair}} := \{C \in \mathcal{C} \mid |C| = 2\} = \mathcal{C} \backslash \mathcal{C}_{\text{sing}}$
$\mathcal{C}'_{\text{sing}} := \varnothing$
while $|\mathcal{C}_{sing}| \neq 0$ **do**
  wähle $C \in \mathcal{C}_{\text{sing}}$ und $\mathcal{C}_{\text{sing}} := \mathcal{C}_{\text{sing}} \backslash \{C\}$, $\mathcal{C}'_{\text{sing}} := \mathcal{C}'_{\text{sing}} \cup \{C\}$
  for $D \in \mathcal{C}_{pair}$ **do**
    $\mathcal{C}_{\text{sing}} := \mathcal{C}_{\text{sing}} \cup \{\text{Res}_L(C, D) \mid \{C, D\} \text{ kann entlang } L \text{ resolviert werden}\}$
  end
end

Algorithm 2: Ausschöpfen von (2.1)

Es gibt nur begrenzt viele Singletons die durch Resolution entstehen können. $\mathcal{C}_{\text{sing}}$ wird immer leerer und es besteht eine immer geringere Chance, in einer while-Schleifen-Instanz, Singletons durch Resolution zu erhalten. Auch die for-Schleife ist polynomiell beschränkt.

for $C \in \mathcal{C}'_{sing}$ **do**
  for $D \in \mathcal{C}'_{sing} \backslash \{C\}$ **do**
    if $\varnothing \in \{\text{Res}_L(C, D) \mid \{C, D\}$ *kann entlang* L *resolviert werden*$\}$ **then**
      return nein
    end
  end
end
return ja

Algorithm 3: Ausschöpfen von (1)

Die Laufzeit ist hier beschränkt durch $\mathcal{O}(|\mathcal{C}_{\text{sing}}|^2)$. ✓ □

Aufgabe 2. Sei $\mathcal{C}$ eine Klauselmenge, sei $C \in \mathcal{C}$ eine Klausel und $L_1, L_2 \in C$. Sei $C_0 = C \backslash \{L_1, L_2\}$. Finden Sie, unter Zuhilfenahme eines neuen Atoms p, Klauseln $C_1, \ldots, C_n$ die jeweils höchstens drei Literale enthalten so dass $\mathcal{C}$ erfüllbar ist genau dann wenn

$$(\mathcal{C} \backslash \{C\}) \cup \{C_0 \cup \{p\}, C_1, \ldots, C_n\}$$

erfüllbar ist. Wir betrachten das folgende Entscheidungsproblem:

3SAT

Eingabe: Klauselmenge $\mathcal{C}$ in der jede Klausel höchstens 3 Literale enthält
Frage: Ist $\mathcal{C}$ erfüllbar?

Zeigen Sie dass 3SAT **NP**-vollständig ist.

Lösung. Zunächst ein kleines ...

Lemma.

- Wenn $C_1 \subseteq C_2 = C$ Klauseln sind, dann gilt, für alle Belegungen $b : \mathrm{L}(\{C_1, C_2\}) \to \{0,1\}$, dass

$$\hat{b}(C_1) \leqslant \hat{b}(C_2) \quad \text{und} \quad \hat{b}(C) = 1 \iff \exists\, L \in C : \hat{b}(L) = 1.$$

- Wenn $\mathcal{C}_1 \subseteq \mathcal{C}_2 = \mathcal{C}$ Klauselmengen sind, dann gilt für alle Belegungen $b : \mathrm{L}(\mathcal{C}_1 \cup \mathcal{C}_2) \to \{0,1\}$, dass

$$\hat{b}(\mathcal{C}_1) \geqslant \hat{b}(\mathcal{C}_2) \quad \text{und} \quad \hat{b}(\mathcal{C}) = 1 \iff \forall\, C \in \mathcal{C} : \hat{b}(C) = 1.$$

Seien $L_3, \ldots, L_m$ die restlichen Literale aus C, d.h.

$$C = \{L_1, \ldots, L_m\} \supset C_0 = \{L_3, \ldots, L_m\}.$$

Sei weiters

Was sind hier gezeigt? Was ist $C_1, \ldots, C_n$?

$$\mathcal{C}' := (\mathcal{C} \backslash \{C\}) \cup \{C_0 \cup \{p\}, C_1, \ldots, C_n\}.$$

„ $\Longleftarrow$ “: Sei $\mathcal{C}'$ erfüllbar. Dann gibt es eine Belegung $b : \mathrm{L}(\mathcal{C}') \to \{0,1\}$, sodass $\hat{b}(\mathcal{C}') = 1$. Insbesondere gilt also

$$\hat{b}(\mathcal{C} \backslash \{C\}) = 1, \quad \text{und} \quad \hat{b}(C_0 \cup \{p\}) = \hat{b}(C_1) = \cdots = \hat{b}(C_n) = 1.$$

1. Fall $(b(p) = 0)$:
   Angenommen, $\hat{b}(C_0) = 0$, dann wäre

$$1 = \hat{b}(C_0 \cup \{p\}) = \underbrace{\hat{b}(C_0)}_{0} \vee \underbrace{b(p)}_{0} = 0.$$

   Also ist $\hat{b}(C_0) = 1$. Damit ist auch $\hat{b}(C) = 1$.

2

2. Fall ($b(p) = 1$):
   Seien

$$C_1 := C' \cup \{\neg p\}, \quad C' \subseteq C, \quad |C'| = 1, 2.$$

   Dann gilt

$$1 \geqslant \hat{b}(C) \geqslant \hat{b}(C') = \hat{b}(C') \vee \underbrace{\hat{b}(\neg p)}_{0} = \hat{b}(C_1) = 1.$$

Insgesamt bekommen wir also $\hat{b}(\mathcal{C}) = 1$.

„ $\Longrightarrow$ “: Sei nun $\mathcal{C}$ erfüllbar. Dann gibt es eine Belegung $b : \mathrm{L}(\mathcal{C}) \to \{0, 1\}$, sodass $\hat{b}(\mathcal{C}) = 1$. Insbesondere gilt also

$$\hat{b}(\mathcal{C} \backslash \{C\}) = 1 \quad \text{und} \quad \hat{b}(C) = 1.$$

1. Fall ($\hat{b}(C_0) = 0$):
   Damit $\hat{b}(\mathcal{C}') = 1$, muss

$$1 \stackrel{!}{=} \hat{b}(C_0 \cup \{p\}) = \underbrace{\hat{b}(C_0)}_{0} \vee \hat{b}(p) = b(p).$$

   Außerdem gilt $\hat{b}(\{L_1, L_2\}) = 1$, sonst wäre

$$1 = \hat{b}(C) = \underbrace{\hat{b}(C_0)}_{0} \vee \underbrace{\hat{b}(\{L_1, L_2\})}_{0} = 0.$$

   Damit $\hat{b}(C) = 1$, muss aber auch

$$1 \stackrel{!}{=} \hat{b}(C_1) = \hat{b}(C') \vee \underbrace{\hat{b}(\neg p)}_{0} = \hat{b}(C').$$

   Wir wählen also $C' := \{L_1, L_2\}$.

2. Fall ($\hat{b}(C_0) = 1$):
   Wir können also getrost $b(p) := 0$ wählen und trotzdem gilt

$$\hat{b}(C_0 \cup \{p\}) = \underbrace{\hat{b}(C_0)}_{1} \vee \underbrace{\hat{b}(p)}_{0} = 1.$$

   Außerdem gilt

$$1 \geqslant \hat{b}(C_1) = \hat{b}(C') \vee \underbrace{\hat{b}(\neg p)}_{1} \geqslant 1.$$

Insgesamt bekommen wir also

$$1 \geqslant \hat{b}(\mathcal{C}') = \underbrace{\hat{b}(\mathcal{C}\backslash\{C\})}_{1} \wedge \underbrace{\hat{b}(C_0 \cup \{p\})}_{1} \wedge \underbrace{\hat{b}(C_1)}_{1} = 1.$$

Was ist C_n?

Fun Fact: Für alle Belegungen $b : \mathrm{L}(\mathcal{C}') \to \{0,1\}$ gilt

$$\hat{b}(C_1) = \hat{b}(\{L_1, L_2\} \cup \{p\}) = \hat{b}(\neg p \vee (L_1 \vee L_2)) = \hat{b}(p \to (L_1 \vee L_2)),$$

und

$$\hat{b}(C_0 \cup \{p\}) = \hat{b}(\{L_3, \ldots, L_m\} \cup \{p\}) = \hat{b}(\neg(\neg p) \vee (L_3 \vee \cdots \vee L_m)) = \hat{b}(\neg p \to (L_3 \vee \cdots \vee L_m)).$$

Nun können wir das soeben Gezeigte auf alle (endlichen) Klauseln C, einer beliebigen (endlichen) Klauselmenge $\mathcal{C}$, mit $|C| > 3$, anwenden. In polynomieller Laufzeit können wir für $\mathcal{C}$ also eine erfüllbarkeitsäquivalente Klauselmenge $\mathcal{C}'$ finden, sodass für alle $C \in \mathcal{C}'$ gilt $|C| \leqslant 3$.
Nun ist

- laut Satz 4.4, SAT =: L **NP**-vollständig,
- laut Satz 4.2, $L \in$ **NP** und damit auch $L' := 3\text{SAT} \in$ **NP**, und
- laut dem soeben Gezeigten, L auf L' polynomiell reduzierbar, i.Z. $L \leqslant_{\mathrm{p}} L'$.

Laut Lemma 4.2, ist also auch L' **NP**-vollständig. (⌣) □

Aufgabe 3. Sei $G = (V, E)$ ein ungerichteter Graph. Eine *Knotenüberdeckung* von G ist eine Menge $V' \subseteq V$ so dass

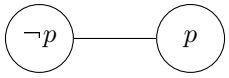
$$\{u, v\} \in E \implies u \in V' \text{ oder } v \in V'.$$

Das Knotenüberdeckungsproblem ist:

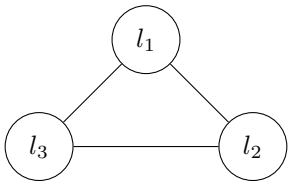
Knotenüberdeckung

Eingabe: ein endlicher ungerichteter Graph G, ein $k \in \mathbb{N}$
Frage: Ist $\mathcal{C}$ erfüllbar?

Zeigen Sie dass Knotenüberdecknung **NP**-vollständig ist.
Hinweis: Reduzieren Sie 3SAT *auf Knotenüberdecknung indem Sie zur Darstellung eines Atoms p den Baustein*



verwenden und zur Darstellung einer Klausel $l_1 \vee l_2 \vee l_3$ den Baustein



Lösung. Sei $\mathcal{C}$ eine Klauselmenge mit Klauseln $C \in \mathcal{C}$, sodass $|C| \leqslant 3$. Betrachte den Graphen $G = (V, E)$ mit $V = V_{\text{Atom}} \cup V_{\text{Klausel}}$. Für jedes Atom $p \in \mathrm{L}(\mathcal{C})$ seien $(p, \_, \_)$, $(\neg p, \_, \_)$ durch Kanten aus E verbundene (Atom-)Knoten aus V_{Atom} (Atom-Bauteil). Für jede Klausel

- $\{L\} = C \in \mathcal{C}$ seien $(L, C, 1), (L, C, 2), (L, C, 3)$
- $\{L_1, L_2\} = C \in \mathcal{C}$ seien $(L_1, C, 1), (L_2, C, 1), (L_2, C, 2)$ oder $(L_1, C, 1), (L_1, C, 2), (L_2, C, 1)$
- $\{L_1, L_2, L_3\} = C \in \mathcal{C}$ seien $(L_1, C, 1), (L_2, C, 1), (L_3, C, 1)$

durch Kanten aus E verbundene (Klausel-)Knoten aus V_{Klausel} (Klausel-Bauteil). Es seien zwei (verschiedene) Knoten aus V_{Atom} bzw. V_{Klausel} ebenfalls durch eine Kante aus E verbunden, wenn ihre ersten Komponenten gleich sind. Sonst mögen V und E nichts Weiteres enthalten.

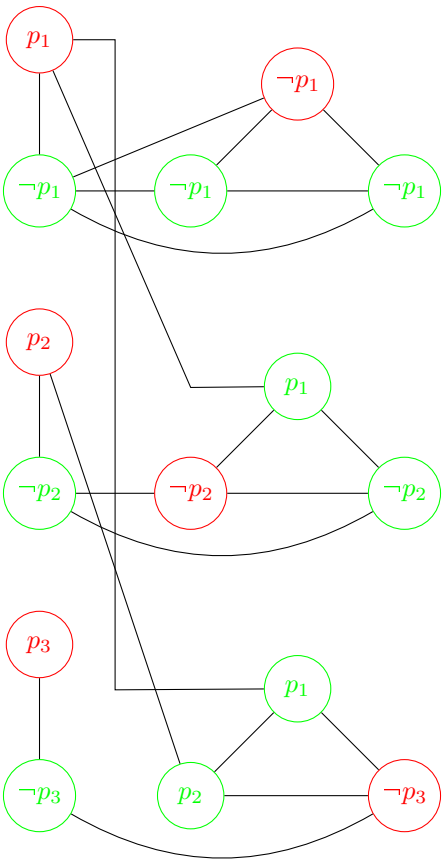
Beispiel. Betrachte die Klauselmenge

$$\mathcal{C} := \{\{\neg p_1\}; \{p_1, \neg p_2\}; \{p_1, p_2, \neg p_3\}\}.$$

Diese wird erfüllt durch die Belegung

$$b : \begin{cases} p_1 \mapsto 0, \\ p_2 \mapsto 0, \\ p_3 \mapsto 0. \end{cases}$$

Im folgenden Graphen sind die roten Kanten nicht in der Knotenüberdeckung enthalten, die grünen allerdings schon.



„ $\Longrightarrow$ “: Sei $\mathcal{C}$ erfüllbar und $b : \mathrm{L}(\mathcal{C}) \to \{0,1\}$ eine Belegung mit $\hat{b}(\mathcal{C}) = 1$. Weil alle Klauseln $C \in \mathcal{C}$ ja $\hat{b}(C) = 1$ erfüllen, also jeweils für mindestens ein $L \in C$ gilt $\hat{b}(L) = 1$, können wir eine Auswahlfunktion f betrachten:

$$f : \mathcal{C} \to \bigcup \mathcal{C} : C \mapsto f(C) \in \left\{ L \in C \mid \hat{b}(L) = 1 \right\} \neq \varnothing.$$

Seien

$$V'_{\text{Atom}} := \left\{ (L, \_, \_) \in V_{\text{Atom}} \mid \hat{b}(L) = 1 \right\},$$
$$V'_{\text{Klausel}} := \{ v \in V_{\text{Klausel}} \mid C \in \mathcal{C}, v \neq (f(C), C, 1) \},$$

und $V' := V'_{\text{Atom}} \cup V'_{\text{Klausel}}$. Dann ist V' eine Knotenüberdeckung von V mit $|V'| \leqslant k = |\mathrm{L}(\mathcal{C})| + 2|\mathcal{C}|$.

1. Sei $e = \{v_1, v_2\} \in E$ eine Kante zwischen zwei Atom-Knoten $v_1 = (p, \_, \_), v_2 = (\neg p, \_, \_) \in V_{\text{Atom}}$. Wenn $b(p) = 1$, dann ist $v_1 \in V'_{\text{Atom}} \subseteq V'$, sonst ist $b(\neg p) = 1$ und $v_2 \in V'_{\text{Atom}} \subseteq V'$.

2. Sei $e = \{v_1, v_2\} \in E$ eine Kante zwischen zwei Klausel-Knoten $v_1, v_2 \in V_{\text{Klausel}}$. Es kann nicht: $v_1 \notin V'_{\text{Klausel}}$ und $v_2 \notin V'_{\text{Klausel}}$ sein, weil jeweils genau ein Knoten aus jedem Klausel-Bauteil fehlt. Wende De Morgan an.

3. Sei $e = \{v_1, v_2\} \in E$ eine Kante zwischen einem Atom-Knoten $v_1 = (L, \_, \_) \in V_{\text{Atom}}$ und Klausel-Knoten $v_2 = (L, C, i) \in V_{\text{Klausel}}$. Angenommen, $v_1, v_2 \notin V'$, dann gilt

   $$v_1 \notin V'_{\text{Atom}}, \quad \text{d.h.} \quad \hat{b}(L) = 0,$$

   und

   $$v_2 \notin V'_{\text{Klausel}}, \quad \text{d.h.} \quad L = f(C), \quad i = 1,$$
   $$\text{also} \quad \hat{b}(L) = \hat{b}(f(C)) = 1.$$

   Widerspruch!

„ $\Longleftarrow$ “: Sei V' eine Knotenüberdeckung von V mit $|V'| \leqslant k = |\mathrm{L}(\mathcal{C})| + 2|\mathcal{C}|$. Weil die Klausel-Knoten in den jeweiligen Klausel-Bausteinen alle verbunden sind, müssten jeweils zwei davon in V' liegen, insgesmt also $2|\mathcal{C}|$. Weil die Atom-Knoten in den jeweiligen Atom-Bausteinen alle verbunden sind, muss jeweils einer davon in V' liegen, insgesmt also $|\mathrm{L}(\mathcal{C})|$. Das sind alle; insgesamt insgesamt also $k = |V'|$.
Sei

$$b : \mathrm{L}(\mathcal{C}) \to \{0,1\} : p \mapsto \begin{cases} 1, & (p, \_, \_) \in V', \\ 0, & \text{sonst.} \end{cases}$$

Sei $C \in \mathcal{C}$ eine beliebige Klausel und $v_2 := (L, C, i) \notin V'$, insbesondere also $L \in C$. Laut Konstruktion von G, gibt es eine Kante $e = \{v_1, v_2\}$, wobei $v_1 = \{L, \_, \_\} \in V_{\text{Atom}}$ ein Atom-Knoten ist. Weil V' eine Knotenüberdeckung ist, $e = \{v_1, v_2\} \in E$, und $v_2 \notin V'$, muss $v_1 \in V'$. Per Definitionem von b, gilt daher

$$1 = \hat{b}(L) \leqslant \hat{b}(C) \leqslant 1.$$

Nachdem das also für alle Klauseln gilt, folgt

$$\hat{b}(\mathcal{C}) = \bigwedge_{C \in \mathcal{C}} \underbrace{\hat{b}(C)}_{1} = 1.$$

Nun ist

- laut Aufgabe 2, $L := 3\text{SAT}$ **NP**-vollständig,
- laut Satz 4.3, $L' :=$ KNOTENÜBERDECKNUNG $\in$ **NP**, und
- laut dem soeben Gezeigten, L auf L' polynomiell reduzierbar, i.Z. $L \leqslant_{\text{p}} L'$.

Laut Lemma 4.2, ist also auch L' **NP**-vollständig. □