Übung zur Vorlesung

Theoretische Informatik II

Prof. Christoph Kreitz, Dipl. Math. Eva Richter

Universität Potsdam, Theoretische Informatik — Wintersemester 2003/04

Blatt 7 — Abgabetermin: Musterlösung steht ab 30.01.04 im Netz

---

Dieses Übungsblatt ist das letzte in diesem Semester und wird nicht mehr korrigiert. Das heißt, daß es nicht mehr in die Bewertung eingeht. Um jedoch auf die Klausur vorzubereitet zu sein, sollten Sie auch das Thema NP-Vollständigkeit ausreichend geübt haben. Als Kontrolle für Sie wird in ca. 10 Tagen eine Musterlösung im Netz bereitgestellt.

Aufgabe 7.1

2-SAT sei der Spezialfall des SAT-Problems, das in der Vorlesung vorgestellt wurde, mit genau zwei Literalen pro Klausel. Es soll angenommen werden, daß gilt $\mathcal{NP} \neq \mathcal{P}$.

Beweisen oder widerlegen Sie: 2-SAT ist $\mathcal{NP}$-vollständig.

Aufgabe 7.2

Ein Monom ist eine Konjunktion von Literalen. Zeigen Sie: das Problem, für eine (endliche) Disjunktion von Monomen zu entscheiden, ob es eine Eingabe gibt, für die alle Monome 0 berechnen, ist $\mathcal{NP}$-vollständig.

Aufgabe 7.3 (Hausaufgabe)

Ein Eulerkreis in einem ungerichteten Graphen ist ein Kreis, der jede Kante genau einmal enthält. Ist das Entscheidungsproblem, ob ein Graph einen Eulerkreis enthält, $\mathcal{NP}$-vollständig oder in $\mathcal{P}$ enthalten (unter der Annahme $\mathcal{NP} \neq \mathcal{P}$)?

Hinweis: Für die Lösung dieser Aufgabe , ist es von Vorteil die folgenden Begriffe aus der Graphentheorie zu benutzen. Ein Graph G heißt <u>zusammenhängend</u>, wenn jede Ecke über Kanten von jeder anderen Ecke erreichbar ist. Eine Ecke hat einen <u>geraden Grad</u>, wenn die Anzahl der Kanten die in dieser Ecke zusammentreffen gerade ist. Damit ist Teilgraph von G genau dann ein <u>Euler'scher Kreis</u>, wenn er zusammenhängend ist und jede seiner Ecken einen geraden Grad hat.

Lösung 7.1

Wer das Buch von Wegener aufmerksam gelesen hat, wird die Bemerkung gefunden haben, daß im Gegensatz zum SAT-Problem mit genau drei Literalen pro Klausel (3-SAT) das Problem 2-SAT$\in P$ ist. Damit ist 2-SAT natürlich nicht $\mathcal{NP}$-vollständig (wenn $P \neq NP$).

Zum Beweis geben wir eine Prozedur an, die zu einer Formel $C := (c_1, ..., c_m)$ mit $c_i = z_{i1} \vee z_{i2}$ und $z_{ij} \in \{x_1, \overline{x_1}, ..., x_n,$
$olx_n\}$ in polynomieller Zeit eine Belegung der Variablen $x_1, ..., x_n$ findet, welche alle Klauseln aus C erfüllt, oder feststellt, daß keine solche existiert.

Um eine solche Belegung zu finden, müssen wir in jeder Klausel c_i mindestens eines der Literale z_{ij} mit 1 belegen können. Das ist immer dann möglich, wenn wir aus jeder Klausel die Literale so herausgreifen können, daß keine Widersprüche entstehen – d.h. es darf nicht gleichzeitig ein x_j und ein $\overline{x_j}$ herausgegriffen werden. Variablen, die beim Herausgreifen nicht belegt werden, spielen keine Rolle und können mit 1 belegt werden. Ich gebe drei Beispiele:

- $C_1 := \{A \vee B, \overline{B} \vee C\}$ Wähle A und $\overline{B}$ und setze entsprechend $A = C = 1$, $B = 0$.
- $C_2 := \{A \vee B, \overline{B} \vee A, \overline{A} \vee \overline{B}\}$ Wähle A und $\overline{B}$ und setze entsprechend $A = 1$, $B = 0$.
- $C_3 := \{A \vee B, \overline{B} \vee A, \overline{A} \vee B, \overline{A} \vee \overline{B}\}$ Hier ist keine Belegung mehr möglich.

Ein naives Verfahren würde aus jeder Klausel beliebig ein Literal herausgreifen und dann testen, ob es Widersprüche gibt. Dieses Verfahren ist jedoch exponentiell, da man im Extremfall alle Kombinationsmöglichkeiten durchtesten muß. Man kann dies jedoch dadurch optimieren, daßs man iterativ vorgeht und *zuerst* die Widersprüche abschneidet:

1. Wir beginnen mit der Klausel c_1 und wählen als erstes Literal z_{11} und merken uns, daß z_{12} eventuell später noch betrachtet werden muß.

2. Wir suchen nun zum Literal z_{11} eine Klausel c_i, in der eines der beiden Literale – z.B. z_{i1} – zu z_{11} *komplementär* ist (d.h. $z_{11} = \overline{z_{i1}}$). Wenn wir dies finden, dann können wir darauf verzichten, Kombinationen von Literalen zu untersuchen, die z_{11} und z_{i1} enthalten.

   Stattdessen verfolgen wir die Teilkombination $[z_{11}, z_{i2}]$ weiter und suchen nun eine *neue* Klausel, die ein zu z_{i2} (oder zu z_{11}) komplementäres Literal enthält. Dadurch können wir weitere Kombinationsmöglichkeiten frühzeitig abschneiden.

   Mit einer Iteration dieses *Extensionsschritts* bauen wir schrittweise eine (Teil-)kombination von Literalen auf (z.B. $[z_{11}, z_{i2}, z_{j2}, \ldots]$), die nicht widersprüchlich ist. Wir wissen gleichzeitig, daß wir die anderen Literale der bereits untersuchten Klauseln nicht mehr in Betracht ziehen müssen.

3. Das Verfahren ist beendet, wenn *alle* Klauseln betrachtet wurden. Die in der aufgebauten Kombination von Literalen genannten Variablen werden – je nachdem, ob sie negiert vorkommen oder nicht – mit 0 bzw. 1 belegt. Nicht genannte Variablen werden mit 1 belegt.

4. Ist kein Extensionsschritt ausführbar, weil es keine Klausel mehr gibt, die ein Literal enthält, das komplementär zu einem bereits gewählten Literal ist, dann können wir die bisherige Teilkombination als gesichert annehmen – sie erzeugt auf keinen Fall mehr Widersprüche.

Wir merken uns, daß die Teilkombination bis zu dieser Stelle sicher ist und wählen aus den noch verbliebenden Klauseln eine beliebige Klausel c_k heraus und gehen ähnlich vor wie im Schritt 1: wir ergänzen die Teilkombination um z_{k1} und merken uns, daß *nur noch* z_{k2} eventuell später noch betrachtet werden muß (die Betrachtung von z_{12} ist nicht mehr nötig)

5. Wird bei einem Extensionsschritt festgestellt, daß *alle* Literale der gewählten Klausel komplementär zu einem der in der Teilkombination vorkommenden Literale sind, dann kann diese Teilkombination nicht mehr zum Ziel führen. Wir die Suche an dieser Stelle abbrechen. Allerdings könnte es sein, daß wir eine Kombination, die mit einem anderen Literal beginnt, noch weiterverfolgen müssen.

   Ist noch ein Literal – z.B. z_{k2} – als unbearbeitet vermerkt, dann entfernen wir aus der Teilkombination von hinten alle Literale bis zu der Stelle, die als gesichert galt, ersetzen z_{k1} durch z_{k2} und beginnen neu. Der Vermerk, daß z_{k2} noch unbearbeitet ist, wird gelöscht.

   Gibt es kein unbearbeitetes Literal mehr, dann ist *jede* Kombination widersprüchlich (enthält mindestens ein komplementäres Paar von Literalen) – die Formel ist unerfüllbar.

Diese Vorgehensweise, die sich leicht auf mehr als 2 Literale pro Klausel erweitern läßt, bringt eine erhebliche Effizienzsteigerung. Im Allgemeinfall bleibt sie aber exponentiell, da in jedem Extensionsschritt weitere Literale als unbearbeitet vermerkt werden müssen. Bei 3 Literalen je Klausel können wir in jedem Schritt ein Literal streichen, das zweite verfolgen wir weiter, aber das dritte muß in eine Liste von Alternativen eingetragen werden. Beim Rücksetzen (Schritt 5) müssen *alle* Alternativen verfolgt werden, was schon bei 3-SAT zu einem Suchbaum der Größe 2^m führt.

Bei der Beschränkung auf 2 Literale je Klausel jedoch gibt es maximal *eine* Alternative, zu der zurückgesetzt werden muß und von der aus neu begonnen wird. Von diesem Neubeginn ausgehend kann im allenfalls durch den Schritt 4 eine *neue* Alternative erzeugt werden. In diesem Fall kann aber alles, was bisher konstruiert wurde, als gesichert angesehen werden (denn es gibt ja keine Möglichkeiten mehr für Widersprüche). Das bedeutet, daß wir bei eventuellem erneutem Rücksetzen mindestens eine Extension weniger ausführen müssen.

Im schlimmsten Fall (bei der Formel $(x_1 \vee x_0) \wedge (x_2 \vee \overline{x_1}) \wedge (x_3 \vee \overline{x_2}) \wedge \ldots (\overline{x_{m-1}} \vee \overline{x_m}))$ verfolgen wir jeweils alle Klauseln bis zum Ende, setzen zurück, finden nichts komplementäres, wählen eine neue Klausel und ein neues Literal, verfolgen wieder alle Klauseln bis zum Ende usw. Dies sind in etwa $m^2/2$ Extensionsschritte, die ausgeführt werden müssen. Da das Suchen nach komplementären Literalen ebenfalls maximal $2m$ Schritte erfordert, arbeitet der Algorithmus in kubischer Zeit.

Beispiele:

- $C_1 := \{A \vee B, \overline{B} \vee C\}$ Wähle $z_{11} = A$, dann beliebig $z_{21} = \overline{B}$. Fertig
- $C_2 := \{A \vee B, \overline{B} \vee A, \overline{A} \vee \overline{B}\}$ Wähle $z_{11} = A$, dazu komplementär ist $z_{31} = \overline{A}$. Wir verfolgen $[z_{11}, z_{32}] = [A, \overline{B}]$ weiter und finden nichts komplementäres mehr. Die Wahl der zweiten Klausel liefert die Kombination $[A, \overline{B}, \overline{B}]$
- $C_3 := \{A \vee B, \overline{B} \vee A, \overline{A} \vee B, \overline{A} \vee \overline{B}\}$ Wähle $z_{11} = A$, dazu komplementär ist $z_{31} = \overline{A}$. Zu $z_{32} = B$ komplementär ist $z_{21} = \overline{B}$. Zu $z_{22} = A$ komplementär ist $z_{41} = \overline{A}$. Leider ist $z_{42} = \overline{B}$ komplementär zu $z_{32} = B$ und damit ist diese Kombination widersprüchlich. Dasselbe erleben wir, wenn wir $z_{12} = B$ verfolgen. Damit gibt es keine Belegung.

Lösung 7.2 *Ziel dieser Aufgabe ist es, NP-vollständige Probleme wiederzuerkennen.*

Das Problem ist sehr stark verwandt mit dem Erfüllbarkeitsproblem SAT, welches nach der Erfüllbarkeit einer Konjunktion von Disjunktionen fragte. Eine solche ist genau dann erfüllbar, wenn die duale Form – eine Disjunktion von Konjunktionen der negierten Literale – unerfüllbar ist. Diese duale Form ist genau eine Disjunktion von Monomen.

KommentarEs sollte bekannt sein, daß nach den DeMorgan'schen Regeln die Negation einer Formel in KNF genau die DNF ist, bei der alle Literale durch ihr Negat ersetzt werden. Ebenso sollte klar sein, daß eine Belegung der Variablen eine Formel genau dann erfüllt, wenn sie in der Negation der Formel zu einer 0 führen, d.h. genau dann, wenn alle Monome eine 0 liefern.

Da die Äquivalenz so offensichtlich ist, ist die Reduktion von SAT auf dieses Problem simpel: Ersetze jedes Literal durch sein Negat, jedes Monom durch eine (disjunktive) Klausel und die Diskunktion der Monome durch die Klauselmenge. Syntaktisch ist das nur eine Ersetzung der logischen Symbole durch andere und geht in linearer Zeit.

Da das Problem in NP liegt (rate und verifiziere die Belegung) und das NP-vollständige SAT-Problem hierauf reduzierbar ist, folgt die Behauptung.

Lösung 7.3

Die Lösung des Problems benötigt einige graphentheoretische Vorüberlegungen, die wir nicht im Detail beweisen:

Wenn ein ein Graph einen Eulerkreis enthält, dann muß *jede Ecke einen geraden Grad haben* (d.h. sie ist mit einer geraden Anzahl von anderen Ecken verbunden). Ansonsten könnte ein Kreis der zu einer Ecke führt, diese nicht wieder verlassen. Ein Graph, der einen Eulerkreis enthält, muß natürlich auch *zusammenhängend* sein, d.h. jede Ecke muß über Kanten von jeder anderen Ecke erreichbar sein.

Man kann nun umgekehrt zeigen, daß die Bedingung "*zusammenhängend und jede Ecke hat einen geraden Grad*" nicht nur notwendig, sondern auch hinreichend ist. Der Grund ist, daß man zwei Eulerkreise, die sich in einem Punkt berühren, zu einem neuen zusammensetzen kann wie zwei Kreise zu einer Acht. Formal beweist man dies durch (Doppel-)Induktion über die Anzahl m der Ecken mit maximalem Grad $2n$. VPP

Wir müssen also nur für einen Graphen $G = (V, E)$ testen, ob er zusammenhängend ist und ob alle Ecken geraden Grad haben. Letzteres geht in quadratischer Zeit relativ zur Länge der Eingabe (zu jeder Ecke zähle durch, wie oft sie in der Kantenmenge genannt wird).

Um Zusammenhang zu prüfen, reicht es, zu testen, ob alle Ecken von *einer* Ecke $v_0 \in V$ erreichbar sind. Dazu bauen wir iterativ Mengen $V_i \subseteq V$ auf, welche die "in i Schritten von v_0 erreichbaren Ecken" beschreiben. Es ist $V_0 := \{v_0\}$, $V_{i+1} := V_i \cup \{v' | \exists v \in V_i . \{v, v'\} \in E\}$
Jede Ecke ist in maximal $|V|$ Schritten erreichbar oder niemals (bei größerer Schrittzahl gab es einen Kreis). *Damit ist G zusammenhängend genau dann, wenn $V = V_{|V|}$*.
Die Menge $V_{|V|}$ ist in $O(|V|^4)$ Schritten konstruierbar (bei geschickterer Programmierung auch in $O(|V|^2)$ Schritten).

Insgesamt kostet der Test auf die Existenz eines Eulerkreises (ohne diesen explizit zu konstruieren) also nur polynomial viele Schritte. Damit ist das Problem in P.

KommentarInteressant hieran ist, daß trotz der scheinbaren Ähnlichkeit zum Hamilton-Kreis Problem (alle Kanten statt alle Ecken) keine $\mathcal{NP}$-vollständigkeit vorliegt.