Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 4

Vorlesung Theoretische Grundlagen der Informatik im WS 17/18

Ausgabe 6. Dezember 2017
Abgabe 19. Dezember 2017, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(2 + 2 = 4 Punkte)

Betrachten Sie folgende Probleme[1]:

> Das Problem 2Color ist das Problem Color aus der Vorlesung mit festem Parameter $k = 2$.
>
> Bei dem Problem Max2Color ist ein Graph G und eine natürliche Zahl k gegeben. Es wird gefragt, ob es eine 2-Färbung der Knoten von G gibt, so dass mindestens k Kanten von G Endpunkte in beiden Farben haben.

(a) Zeigen Sie dass 2Color in P ist.

(b) Zeigen Sie dass Max2Color NP-vollständig ist.

Aufgabe 2

(5 Punkte)

In der Vorlesung wurde gezeigt, dass 3Sat NP-vollständig ist. Sie dürfen davon ausgehen, dass dabei niemals eine Variable mehrfach in einer Klausel vorkommt. Sei 4Sat folgendes Problem:

> Sei V Menge an Variablen und $\mathcal{C}$ eine Menge von Klauseln über V, wobei jede Klausel $C \in \mathcal{C}$ aus genau vier unterschiedlichen veroderten Literalen besteht, also z.B. $C = (x_3 \vee \neg x_5 \vee x_6 \vee x_9)$.
> Gibt es eine Wahrheitsbelegung von V, so dass jede Klausel in $\mathcal{C}$ erfüllt ist?

Zeigen Sie, dass 4Sat NP-vollständig ist.

Aufgabe 3

(4 + 1 = 5 Punkte)

Bei der *Quadratischen Programmierung* besteht eine Eingabe aus einer Menge von Ungleichungen mit Polynomen von höchstens Grad Zwei (mit ganzzahligen Koeffizienten) über n reellen Variablen

[1] Vgl. Vorlesung 9, 28.11.2017, Folie 14

$x_1, x_2, \ldots, x_n$. Das Problem QuadProg ist, zu entscheiden, ob sich die Variablen so reellen Zahlen zuweisen lassen, dass alle Ungleichungen erfüllt sind.

Hier ist ein kleines Beispiel:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1^2 + x_2^2 &\leq 1 \\ 9x_1x_2 &\geq 1 \end{aligned}$$

Diese Instanz ist erfüllbar, zum Beispiel durch $x_1 = x_2 = \frac{1}{2}$. Ersetzt man die letzte Ungleichung jedoch durch $x_1x_2 \geq 1$ gibt es keine erfüllende Belegung.

(a) Zeigen Sie, dass QuadProg NP-schwer ist. Benutzen Sie die NP-Schwere von 3Color, die in der Vorlesung bewiesen wurde.

(b) Um zu zeigen, dass QuadProg NP-vollständig ist, müsste man noch zeigen, dass QuadProg $\in$ NP gilt. Dies ist allerdings nicht trivial. Wieso?

Aufgabe 4

(1 + 1 + 1 = 3 Punkte)

Doktor Meta forscht in seinem Labor am Geheimproblem X. Er hat bereits eine polynomielle Transformation von Sat auf X gefunden, die Sat-Instanzen der Größe n auf X-Instanzen der Größe n^3 abbildet. In einem Moment der Genialität gelingt es Doktor Meta nun, eine Laufzeitschranke für Sat von $\Omega(c^n)$ für eine Konstante $c > 1$ zu beweisen.

(a) Was folgt aus der Laufzeitschranke für Sat über die Frage $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$?

(b) Welche Laufzeitschranke ergibt sich für X?

(c) Können Sie mit der Laufzeitschranke für X die Frage $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ beantworten?

Begründen Sie Ihre Antworten!

Aufgabe 5

(2 + 2 + 2 = 6 Punkte)

Wir habens uns ausführlich mit dem Konzept der NP-Vollständigkeit beschäftigt. Der Grund dafür ist, dass die NP-vollständigen Sprachen die „schwersten“ Sprachen in NP sind.

Dieses Konzept von Vollständigkeit lässt sich erweitern. In dieser Aufgabe sollen Sie einen ähnlichen Vollständigskeitsbegriff für eine andere Menge von Sprachen entwickeln. Es sei noch einmal an die Definition einer Transformation erinnert:

Eine *Transformation* einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

- es existiert eine deterministische Turingmaschine M, die f berechnet, und
- für alle $w \in \Sigma_1^*$ gilt $w \in L_1 \iff f(w) \in L_2$.

Wir schreiben dann $L_1 \leq_M L_2$.

Diese Definition gleicht der Definition einer *polynomialen Transformation* aus der Vorlesung[2] bis darauf, dass hier keine polynomiale Laufzeit zur Berechnung von f gefordert wird[3].

Bezeichne SE die Menge der semi-entscheidbaren Sprachen. Eine Sprache L ist SE-schwer, wenn für jede Sprache L' in SE gilt, dass $L' \leq_M L$. Eine SE-schwere Sprache ist also „mindestens so schwer" wie die schwersten Probleme in SE.

(a) Zeigen Sie, dass alle SE-schweren Sprachen nicht entscheidbar sind.

(b) Zeigen Sie, dass wenn L SE-schwer ist und $L \leq_M L'$ gilt folgt, dass L' ebenfalls SE-schwer ist. *Hinweis:* Beweisen Sie, dass $\leq_M$ transitiv ist.

Eine Sprache L ist SE-vollständig, wenn L SE-schwer ist und $L \in$ SE gilt. Damit ist sichergestellt, dass L einerseits mindestens so schwer die schwersten Probleme in SE ist, und andererseits, dass L höchstens so schwer ist wie die schwersten Probleme in SE. Die Universelle Sprache war folgendermaßen definiert:

$$L_u = \{\langle M, w\rangle \mid M \text{ akzeptiert } w\}$$

(c) Zeigen Sie, dass L_u SE-schwer und sogar SE-vollständig ist. Damit haben Sie gezeigt, dass die Universelle Sprache zu den schwersten semi-entscheidbaren Sprachen gehört!

Aufgabe 6 (4 Punkte)

In der vorherigen Aufgabe haben Sie gesehen, dass die Universelle Sprache L_u zu den schwersten semi-entscheidbaren Problemen gehört. In der Vorlesung wurde gezeigt, dass die Diagonalsprache L_d nicht semi-entscheidbar ist. Intuitiv würde man deshalb vielleicht vermuten, dass die Diagonalsprache „schwerer" ist, als die Universelle Sprache. In dieser Aufgabe sollen Sie beweisen, dass ein solcher Schwerevergleich zumindest über Tranformationen nicht möglich ist.

Zeigen Sie dazu, dass $L_d \not\leq_M L_u$ und $L_u \not\leq_M L_d$ gilt, dass die beiden Sprachen also jeweils nicht aufeinander reduzierbar sind.

Aufgabe 7 (3 Punkte)

Die in der Vorlesung definierte nichtdeterministische Turingmaschine wird auch als RV-NTM bezeichnet (Raten/Verifizieren). Eine andere Möglichkeit ist es, NTMs analog zu NEAs zu definieren (vgl. Wegener und Übung):

[2] Siehe z.B. Folie 2 vom 23. November.

[3] Insbesondere gilt also $L_1 \propto L_2 \implies L_1 \leq_M L_2$, die Umkehrung im Allgemeinen aber nicht.

Eine solche alternative nichtdeterministische Turingmaschine (A-NTM) ist definiert wie eine TM, nur ist die Übergangsfunktion δ durch eine zweistellige Relation $\subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\})$ ersetzt, die wir ebenfalls δ nennen.

Die Arbeitsweise einer A-NTM ist die folgende. Wenn die A-NTM im Zustand $q \in Q$ das Zeichen $a \in \Gamma$ liest, ist für jedes $((q, a), (q', a', d)) \in \delta$ der Rechenschritt möglich, den eine DTM für $\delta(q, a) = (q', a', d)$ durchführt. Falls kein Rechenschritt möglich ist, stoppt die A-NTM. Für eine feste Eingabe x können offensichtlich viele Rechenwege möglich sein.

Eine A-NTM $\mathcal{M}$ akzeptiert eine Eingabe x, falls es mindestens einen Rechenweg von $\mathcal{M}$ gibt, der in einen akzeptierenden Endzustand führt. Die von $\mathcal{M}$ erkannte Sprache $L = L(\mathcal{M})$ besteht aus allen Worten, die $\mathcal{M}$ akzeptiert.

Die Rechenzeit für eine Eingabe $x \in L(\mathcal{M})$ ist gleich der Anzahl der Rechenschritte auf einem kürzesten akzeptierenden Rechenweg. Die Zeitkomplexität $T_{\mathcal{M}}(n)$ ist das Maximum der Rechenzeiten für alle Eingaben aus $L(\mathcal{M})$ der Länge n, und 1 falls es keine solche Eingabe gibt.

Die Klasse ANP ist die Klasse aller Probleme, die von einer solchen A-NTM in polynomieller Zeit entschieden werden können. In der Übung wurde/wird[4] gezeigt, dass $\mathsf{ANP} \subseteq \mathsf{NP}$ gilt. Zeigen Sie: $\mathsf{NP} \subseteq \mathsf{ANP}$.

[4] am 12.12.2017