Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Sascha Böhme, Lars Noschinski

Sommersemester 2011
Lösungsblatt 11
1. August 2011

Einführung in die Theoretische Informatik

Hinweis: *Bitte beachten Sie unbedingt die Hinweise zum Übungsablauf und zu den Aufgabentypen auf der THEO-Website (`http://theo.in.tum.de/`).*

Wenn die Zusatzaufgaben abgegeben und Punkte dadurch erworben werden, dann werden diese Punkte der zweiten Semesterhälfte gutgeschrieben. Die 40% Regelung bezieht sich dabei stets auf die normalen Hausaufgaben. Die Zusatzaufgaben können also dazu benutzt werden, fehlende Punkte auszugleichen.

Rückgabe der korrigierten Hausaufgaben: *Eingereichte Lösungen der Hausaufgaben können am Montag, dem 8. August 2011, in der Zeit von 10:00 bis 12:00 und am Dienstag, dem 9. August 2011, zwischen 14:00 und 16:00 jeweils im Raum MI 01.09.011 (Glaskasten) abgeholt werden.*

Hausaufgabe 1 (6 Punkte)

Sei IF die Menge aller aussagenlogischen Formeln, die ausschließlich mit den Konstanten 0 und 1, logischen Variablen x_i mit $i \in \mathbb{N}$ und der Implikation $\rightarrow$ als Operationszeichen aufgebaut sind, wobei natürlich auch Klammern zugelassen sind. Beachten Sie, dass $x_i \rightarrow x_j$ die gleiche Wahrheitstafel wie $(\neg x_i) \vee x_j$ hat.

Wie betrachten das Problem $ISAT$:

Gegeben: $F \in IF$

Problem: Ist F erfüllbar, d.h., gibt es eine Belegung der Variablen mit Konstanten 0 oder 1, so dass F den Wert 1 annimmt?

Zeigen Sie: $ISAT$ ist NP-vollständig. Sie dürfen dazu benutzen, dass das SAT-Problem NP-vollständig ist.

Lösungsvorschlag

- $ISAT \leq_p SAT$ ($ISAT$ ist in NP):

  Sei f die Abbildung, die in jeder Formel F aus IF jedes Vorkommen der Implikation $a \rightarrow b$ durch $\neg a \vee b$ ersetzt. Dann ist f total und in polynomieller Zeit berechenbar. Da SAT in NP liegt und NP nach unten abgeschlossen ist (Lemma 5.16), liegt also auch $ISAT$ in NP.

- $SAT \leq_p ISAT$ ($ISAT$ ist NP-hart):

  Sei f die Abbildung, die in jeder Formel F aus SAT mit Teilformeln a und b jedes Vorkommen

  - der Negation $\neg a$ durch $a \rightarrow 0$,

– der Disjunktion $a \vee b$ durch $(a \to 0) \to b$ und
– der Konjunktion $a \wedge b$ durch $(a \to (b \to 0)) \to 0$

ersetzt. Dann ist f total und in polynomieller Zeit berechenbar. Da SAT bereits NP-hart ist, ist somit auch $ISAT$ NP-hart.

Hausaufgabe 2 (4 Punkte)

Beweisen Sie:

1. P ist abgeschlossen unter Komplement.
2. Das Problem zu entscheiden, ob ein gegebener Graph ein Dreieck enthält, ist in P.

Lösungsvorschlag

1. Sei $A \in \Sigma^*$ in P und sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ eine Turingmaschine, die A in polynomieller Zeit entscheidet, d.h. die charakteristische Funktion χ_A berechnet. Dann ist $M' = (Q \cup \{\overline{q}\}, \Sigma, \Gamma, \delta', q_0, \square, \{\overline{q}\})$ mit

$$\delta'(q,a) = \begin{cases} (\overline{q}, 0, N) & \text{falls } q \in F \text{ und } a = 1 \\ (\overline{q}, 1, N) & \text{falls } q \in F \text{ und } a = 0 \\ \delta(q,a) & \text{sonst} \end{cases}$$

   eine Turingmaschine, die $\overline{A}$ entscheidet, und zwar ebenso in polynomieller Zeit. Daher ist auch $\overline{A}$ in P.

2. Folgender Algorithmus entscheidet dieses Problem:

   > Prüfe für je drei verschiedene Knoten des Graphen, ob sie untereinander verbunden sind.

   Da es höchstens n^3 verschiedene Tripel von Knoten eines Graphen mit insgesamt n Knoten gibt und der Test auf Verbundenheit dreier Knoten in polynomieller Zeit möglich ist, ist das Problem des Dreiecks in einem Graphen in P.

Hausaufgabe 3 (4 Punkte)

Sei A ein entscheidbares Problem und $B \in P$ ein nicht-triviales Problem (d.h. $B \neq \varnothing$ und $B \neq \Sigma^*$). Zeigen Sie: Aus $A \leq B$ können wir keine Aussage über die Komplexität von A treffen.
Hinweis: Betrachten Sie eine Reduktion auf die Menge $\{0, 1\}$.

Lösungsvorschlag

Sei $\chi_A : A \to \{0,1\}$ die charakteristische Funktion von A. Diese ist total und, da A entscheidbar ist, auch berechenbar. Da B nicht-trivial ist, gibt es u, v mit $u \in B$ und $v \notin B$. Sei $f : \{0,1\} \to B$ die Funktion mit $f(0) = v$ und $f(1) = u$, dann ist A mit $f \circ \chi_A$ auf B reduzierbar.
Diese Konstruktion funktioniert für alle berechenbaren Sprachen A, insbesondere auch für solche, die nicht in P (oder NP) liegen. Damit ist keine Komplexitätsaussage über A möglich.

Hausaufgabe 4 (6 Punkte)

Seien $A, B \in \Sigma^*$ Sprachen in NP. Zeigen Sie, dass dann auch $A \cup B$, AB und A^* in NP liegen.
Hinweis: Überlegen Sie sich hierzu, wie geeignete Zertifikate aussehen.

Lösungsvorschlag

Da A und B in NP liegen, können wir annehmen, dass es polynomiell beschränkte Verifikatoren M_A und M_B für A bzw. B gibt. Zum Beispiel für ein $w \in A$ gibt es also ein Zertifikat c, das "beweist", dass $w \in A$ liegt, und das in Polynomialzeit überprüft werden kann. M_A akzeptiert dann das Wort $w\#c$.
Für Vereinigung, Konkatenation und Stern müssen wir uns nun jeweils überlegen, wie Zertifikate aussehen können, und wie sie in Polynomialzeit überprüft werden können.

Vereinigung: Ein Zertifikat für $w \in A \cup B$ besteht aus einer Kennung $b \in \{0, 1\}$, die angibt, ob $w \in A$ oder $w \in B$ gezeigt werden soll. Dann folgt ein Zertifikat für die Zugehörigkeit zu der jeweiligen Menge. Der Verifikator überprüft nun, welcher der beiden Fälle vorliegt und simuliert dann den Verifikator M_A bzw. M_B.

Wir konstruieren einen Verifikator $M_\cup$, so dass

$$w\#c \in L(M_\cup) \iff (c = 0c_A \wedge w\#c_A \in L(M_A)) \vee (c = 1c_B \wedge w\#c_B \in L(M_B))$$

$M_\cup$ prüft also anhand der Kennung, um was für ein Zertifikat es sich handelt, und startet dann entweder M_A oder M_B. Damit haben wir einen polynomiellen Verifikator für $A \cup B$ konstruiert.

Konkatenation Damit ein Zertifikat für $w \in AB$ möglichst einfach zu überprüfen ist, enthält es am besten auch die Aufteilung des Wortes w in zwei Teilwörter $u \in A$ und $v \in B$. Der Verifikator $M_\times$ muss prüfen, ob diese Zerlegung tatsächlich wieder das Wort ergibt und (anhand von Zertifikaten) ob die Teilwörter in A bzw. B liegen:

$$w\#c \in L(M_\times) \iff c = u\#v\#c_A\#c_B \wedge uv = w \wedge u\#c_A \in L(M_A) \wedge v\#c_B \in L(B)$$

Alternativ wäre es auch möglich, die Zerlegung nicht im Zertifikat mitzuliefern. Der Verifikator muss dann selbst eine passende Zerlegung finden. Auch das ist aber in Polynomialzeit möglich, da es nur $|w| + 1$ Zerlegungen gibt, die man alle durchprobieren kann.

Stern Wir benutzen die zweite Idee der Konkatenation hier erneut. Ein Zertifikat c für $w \in A^*$ besteht aus einer Liste von Zertifikaten $c_1, \ldots, c_k$, die angeben, dass Wörter $w_1, \ldots, w_k$ in A liegen mit $w_1 \cdots w_k = w$. Der Verifikator für M_* muss also für alle Zerlegungen für w in k Teilwörter anhand der Zertifikate prüfen, ob diese Teilwörter in A liegen:

$$\begin{aligned} w\#c \in L(M_*) \iff & c = c_1\# \cdots \#c_n \wedge \\ & (\exists w_1, \ldots, w_n.\ w = w_1 \cdots w_n \wedge \\ & (\forall i \in \{1, \ldots, n\}.\ w_i\#c_i \in L(M_A))) \end{aligned}$$

Das Finden der geeigneten Zerlegung wiederum ist in Polynomialzeit möglich, da es für ein Wort der Länge n höchstens n^k Zerlegungen in k Teile gibt, die man alle durchprobieren kann.

Zusatzaufgabe 1 (4 Zusatzpunkte)

Zeigen Sie, dass die folgenden Probleme für Paare von Turingmaschinen unentscheidbar sind:

1. *EQ*.

   Gegeben: Ein Paar von Turingmaschinen (M, N).

   Problem: Gilt $L(M) = L(N)$?

2. *ECUT*.

   Gegeben: Ein Paar von Turingmaschenen (M, N).

   Problem: Gilt $L(M) \cap L(N) = \varnothing$?

Lösungsvorschlag

1. Wir zeigen, dass die Sprachgleichheit *EQ* für Turingmaschinen unentscheidbar ist, indem wir H_0 auf *EQ* reduzieren. Sei w_0 die Kodierung einer Turingmaschine mit $L(M_{w_0}) = \{0\}$ und sei f eine Funktion mit $f(w) = (w', w_0)$, wobei $M_{w'}$ eine TM ist, die folgendes macht:

   - Zunächst wird M_w auf der leeren Eingabe simuliert.
   - Wenn M_w terminiert, gibt sie 0 aus und akzeptiert.

   Jetzt gilt $L(M_{w'}) = \{0\} = M_{w_0}$ genau dann, wenn $w \in H_0$, also $w \in H_0$ genau dann, wenn $f(w) \in EQ$. Weiterhin ist f total und berechenbar und damit folgt, dass *EQ* unentscheidbar ist.

2. Wir reduzieren zunächst H_0 auf $\overline{ECUT}$. Da Entscheidbarkeit abgeschlossen unter Komplement ist, ist damit auch *ECUT* unentscheidbar. Sei w_0 die Kodierung einer Turingmaschine mit $L(M_{w_0}) = \{0\}$. Für eine TM-Kodierung w sei f eine Funktion mit $f(w) = (w', w_0)$, wobei $M_{w'}$ eine TM ist, die folgendes macht:

   - Zunächst wird M_w auf der leeren Eingabe simuliert.
   - Wenn M_w terminiert, gibt sie 0 aus und akzeptiert.

   Jetzt gilt $L(M_{w'}) \cap L(M_{w_0}) \neq \varnothing$ genau dann, wenn $w \in H_0$, also $w \in H_0$ genau dann, wenn $f(w) \in \overline{ECUT}$. Weiterhin ist f total und berechenbar und damit folgt, dass $\overline{ECUT}$ unentscheidbar ist. Mit unserer Vorüberlegung folgt auch die Unentscheidbarkeit von *ECUT*.

Zusatzaufgabe 2 (2 Zusatzpunkte)

Wir betrachten das Problem *PARTITION*:

Gegeben: Eine endliche Menge $A \subseteq \mathbb{N}$.

Problem: Gibt es ein $B \subseteq A$, so dass $\sum_{m \in B} m = \sum_{n \in A \setminus B} n$ gilt?

Zeigen Sie, dass *PARTITION* in NP liegt (in der Vorlesung wurde nur die NP-Härte bewiesen).

Lösungsvorschlag

Wir verwenden Satz 5.10 aus der Vorlesung und zeigen stattdessen, dass es einen polynomiell beschränkten Verifikator gibt. Als Zertifikat verwenden wir die Teilmenge B. Der Verifikator summiert jetzt die Element von B bzw. $A \setminus B$ auf und vergleicht die Summen. Dies ist offensichtlich in polynomieller Zeit möglich.

Zusatzaufgabe 3 (4 Zusatzpunkte)

Beweisen oder widerlegen Sie:
Sei M ein polynomiell beschränkter Verifikator für das Problem A. Verwendet M nur Zertifikate, die logarithmisch in der Eingabelänge sind (das heißt, es gibt ein $x > 0$, so dass $|c| \leq x \cdot \log |w|$ gilt für alle $w\#c \in L(M)$), so ist A in P.

Lösungsvorschlag

Sind die Zertifikate logarithmisch in der Eingabelänge, so gibt es nur polynomiell viele Zertifikate (denn es gibt ein c mit $|\Gamma|^{\log l} = c \cdot l$ für alle $l \in \mathbb{N}$). Um zu entscheiden, ob $w \in A$ gilt, können wir also alle Zertifikate aufzählen und mit jedem dieser Zertifikate den Verifikator M aufrufen. Dann gilt $w \in A$ genau dann, wenn der Verifikator eines dieser Zertifikate akzeptiert. Rufen wir polynomiell häufig einen polynomiellen Algorithmus auf, bleibt die Gesamtlaufzeit polynomiell; also liegt A in P.

Quiz 1

Beantworten Sie kurz die folgenden Fragen:

1. Seien A, B Sprachen mit $A \leq_p B$ und B NP-vollständig. Gilt dann A NP-vollständig?
2. Ist $time_M$ für jede deterministische Turingmaschine M berechenbar?
3. PSPACE ist die Klasse all jener Probleme, die eine DTM mit polynomiell viel Band lösen kann. Gilt $P \subseteq PSPACE$?

Lösungsvorschlag

1. Nein. Es gilt $A \in$ NP, aber diese Reduktion gibt uns keine Aussage darüber, ob A NP-hart ist.
2. Nein, denn dann wäre das Halteproblem entscheidbar.
3. Ja, denn in polynomieller Zeit kann nur polynomiell viel Band beschrieben werden. Ob $PSPACE \subseteq P$ gilt, ist ein offenes Problem.

Tutoraufgabe 1

Eine Teilmenge $A \subseteq V$ von Knoten eines ungerichteten Graphen $G = (V, E)$ nennt man Knotenüberdeckung von G, wenn alle Kanten von G einen Endpunkt in A haben. Wir definieren das Problem der KNOTENÜBERDECKUNG wie folgt:

Gegeben: Ein Graph G und eine Zahl k.

Problem: Gibt es eine Knotenüberdeckung von G mit höchstens k Knoten?

Zeigen Sie, dass das Problem der KNOTENÜBERDECKUNG NP-vollständig ist.

Lösungsvorschlag

Bezeichnen wir das Problem der Knotenüberdeckung bei entsprechender Kodierung als $K\ddot{U} \subseteq \Sigma^*$.

- Offenbar ist $K\ddot{U}$ in NP, denn man kann zu gegebenem Problem $w \in K\ddot{U}$, d. h. einem Graph $G = (V, E)$ mit entsprechendem k, in polynomieller Zeit eine Teilmenge von Knoten A mit $|A| = k$ raten und dann prüfen, ob alle Kanten einen Endpunkt in der Knotenmenge A besitzen.
- Wir reduzieren das Cliquen-Problem $CLIQUE$, von dem wir wissen, dass es NP-hart ist (siehe Vorlesung Satz 5.32), auf $K\ddot{U}$, d. h., wir zeigen $CLIQUE \leq_p K\ddot{U}$, wie folgt.

  Sei $G = (V, E)$ ein ungerichteter Graph. Wir nennen eine Menge von Knoten $A \subseteq V$ unabhängig, falls keine zwei Knoten aus A über eine Kante verbunden sind. Wir definieren außerdem das Komplement von G als den Graphen $\overline{G} = (V, \overline{E})$ mit $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v\} \setminus E$.

  Nun gilt für alle $A \subseteq V$ und $\overline{A} = V \setminus A$ offenbar:

$$A \text{ ist eine Knotenüberdeckung von } G \iff \overline{A} \text{ ist unabhängig in } G$$

Weiterhin gilt für alle $A \subseteq V$:

$$A \text{ ist unabhängig in } G \iff A \text{ ist eine Clique in } \overline{G}$$

denn die Komplementbildung eines Graphen beseitigt alle vorhandenen Kanten.

Daher gelten auch folgende Äquivalenzen:

$$\begin{aligned} & A \text{ ist eine Clique in } G \text{ mit } k \text{ Elementen} \\ \iff & A \text{ ist unabhängig in } \overline{G} \text{ mit } k \text{ Elementen} \\ \iff & \overline{A} \text{ ist eine Knotenüberdeckung in } \overline{G} \text{ mit } |V| - k \text{ Elementen} \end{aligned}$$

Die gesuchte polynomiell berechenbare Reduktionsabbildung $f : \Sigma^* \to \Sigma^*$ ist nun im Wesentlichen gegeben durch die Abbildung, die einem Graphen $G = (V, E)$ zusammen mit einem Parameter k den Graphen $\overline{G} = (V, \overline{E})$ zusammen mit dem Parameter $k' = |V| - k$ zuordnet.

Man beachte, dass in f die Variablen A und $\overline{A}$ nicht explizit auftreten, weil diese Variablen durch den Existenzquantor in den Problemen gebunden sind.

Tutoraufgabe 2

Das Stundenplan-Problem, das in der Praxis allen bekannt ist, lässt sich vereinfacht wie folgt formal beschreiben:

Gegeben: Endliche Mengen S (Studierende), V (Vorlesungen) und T (Termine) und eine Relation $R \subseteq S \times V$. Dabei bedeutet $(s, v) \in R$, dass s die Vorlesung v besuchen möchte.

Problem: Gibt es eine Abbildung $f : V \to T$, so dass alle Studierenden einen überschneidungsfreien Stundenplan haben, also

$$(s, v_1) \in R \wedge (s, v_2) \in R \wedge v_1 \neq v_2 \implies f(v_1) \neq f(v_2)$$

1. Zeigen Sie, dass Stundenplan in NP liegt.
2. Zeigen Sie, dass Stundenplan NP-hart ist, indem sie eine polynomielle Reduktion von 3col auf Stundenplan angeben.
3. Geben Sie nun eine Reduktion von Stundenplan auf SAT an, die es erlaubt, das Stundenplanproblem mit Hilfe eines SAT-Solvers zu lösen.

Lösungsvorschlag

1. Um zu zeigen, dass Stundenplan in NP liegt, genügt es zu zeigen, dass es effizient überprüfbare Zertifikate gibt (vgl. Satz 5.10). Ein solches Zertifikat ist offensichtlich die gesuchte Abbildung f, die man z.B. als geordnete Liste von Terminen (einen pro Vorlesung) kodieren kann. Wir müssen uns lediglich klarmachen, dass dieses Zertifikat in Polynomialzeit überprüft werden kann. Das ist der Fall, denn man kann z.B. zur Überprüfung die Stundenpläne aller Studierenden ($|S|$ Stundenpläne mit maximal $|V|$ Einträgen) aufstellen und auf Überschneidungen prüfen.

2. Die polynomielle Reduktion funktioniert wie folgt: Gegeben eine Instanz $G = (V, E)$ von 3COL, konstruieren wir daraus ein Stundenplanproblem (S, V', T, R) wie folgt:

$$\begin{aligned} S &= E \\ V' &= V \\ T &= \{1, 2, 3\} \\ R &= \{(\{v_1, v_2\}, v) \in S \times V \mid v \in \{v_1, v_2\}\} \end{aligned}$$

Die Knoten des Graphen werden also zu Vorlesungen und die drei Farben zu Terminen, die man den Vorlesungen zuordnet. Für jede Kante zwischen zwei Knoten erzeugt man nun einen Studenten, der genau die beiden entprechenden Vorlesungen hören will und somit verhindert, dass sie auf denselben Termin gelegt werden.

Die oben beschriebene Transformation ist offensichtlich in Polynomialzeit berechenbar (es handelt sich ja im Wesentlichen um eine "Umbenennung" der Konzepte). Somit ist das Graphenfärbungsproblem auf das Stundenplanproblem polynomiell reduzierbar:

$$\text{3COL} \leq_p \text{STUNDENPLAN}$$

Da von 3COL bekannt ist, dass es NP-hart ist (Satz 5.38, wurde in der Vorlesung aber nicht bewiesen), haben wir nun gezeigt, dass STUNDENPLAN NP-hart ist, und somit (mit Teil 1) NP-vollständig.

3. Eine einfache Kodierung eines gegebenen Stundenplanproblems (S, V, T, R) in SAT ist folgende:

   - Für jede Vorlesungs-Termin-Kombination benötigen wir eine Boolesche Variable, die genau dann wahr sein soll, wenn die Vorlesung zu diesem Termin stattfindet. Wir erhalten also die Variablenmenge $\{x_v^t \mid v \in V, t \in T\}$.
   - Vorlesungen sollen nicht an mehreren Terminen stattfinden, was wir durch Constraints
   $$F_1 = \bigwedge_{v \in V} \bigwedge_{t \in T} \bigwedge_{\substack{t' \in T \\ t' \neq t}} \neg(x_v^t \wedge x_v^{t'})$$
   ausdrücken.
   - Die Einschänkungen, die sich aus den Wünschen der Studierenden ergeben, beschreiben wir mit
   $$F_2 = \bigwedge_{\substack{(v_1, v_2) \in V \times V \text{ mit} \\ (s, v_1) \in R, (s, v_2) \in R \\ \text{und } v_1 \neq v_2}} \bigwedge_{t \in T} \neg(x_{v_1}^t \wedge x_{v_2}^t)$$
   - Schließlich müssen wir noch sicherstellen, dass jede Vorlesung auch stattfindet, denn sonst wäre der leere Stundenplan immer eine Lösung:
   $$F_3 = \bigwedge_{v \in V} \bigvee_{t \in T} x_v^t$$

Aus einer Belegung, die $F = F_1 \wedge F_2 \wedge F_3$ erfüllt, kann man dann leicht einen Stundenplan ablesen. Die oben beschriebene Codierung ist offensichtlich in Polynomialzeit berechenbar.

Im Prinzip haben wir mit dieser Einbettung in SAT nur noch einmal gezeigt, dass STUNDENPLAN $\in NP$. Im Gegensatz zu Teilaufgabe 1 haben wir nun aber auch einen Weg aufgezeigt, wie man das Problem praktisch angehen kann, nämlich mit der Hilfe von effizienten Lösern für SAT. Das ist häufig schneller und einfacher, als einen problemspezifischen Algorithmus neu zu entwickeln und zu implementieren.