Theoretische Informatik II

Prof. Dr. Christoph Kreitz / Holger Arnold
Universität Potsdam, Theoretische Informatik, Sommersemester 2006

Übung 12 (Version 3) — Abgabetermin: 17.7.2006, 11.00 Uhr

Quiz 12

Markieren Sie die folgenden Aussagen als wahr (w) oder falsch (f).

[ ] Es ist entscheidbar, ob eine platzbeschränkte Turingmaschine auf einer Eingabe terminiert.

[ ] coPSPACE $\neq$ PSPACE.

[ ] Jedes PSPACE-vollständige Problem ist NP-hart.

Aufgabe 12.1

Wo ist der Fehler in folgendem „Beweis“, dass P $\neq$ NP gilt?

„Angenommen, P = NP. Dann ist SAT $\in$ P, d.h. für ein k ist SAT $\in$ TIME(n^k). Weil jede Sprache in NP polynomiell auf SAT reduziert werden kann, ist NP $\subseteq$ TIME(n^k). Folglich gilt auch P $\subseteq$ TIME(n^k). Nach dem Zeithierarchiesatz gibt es eine Sprache in TIME(n^{k+1}), die nicht in TIME(n^k) liegt. Das widerspricht aber P $\subseteq$ TIME(n^k). Folglich muss P $\neq$ NP gelten.“

Aufgabe 12.2

Zeigen Sie die Gültigkeit der folgenden Aussagen.

1. Aus NP $\neq$ coNP folgt P $\neq$ NP.
2. Aus P = PSPACE folgt NP = coNP.

Aufgabe 12.3

Eine Relation $R \subseteq \Sigma^* \times \Sigma^*$ heißt polynomiell beschränkt, wenn es ein Polynom p gibt, so dass $|y| \leq p(|x|)$ für alle $(x, y) \in R$ gilt (da Funktionen eine spezielle Art von Relationen sind, gilt diese Definition entsprechend). Man kann eine Relation als ein Suchproblem betrachten und sagt dann, dass ein Programm M das Problem R löst, wenn für jedes $(x, y) \in R$ gilt,dass $(x, f_M(x)) \in R$ ist (f_M sei die von dem Programm M berechnete Funktion). Beweisen Sie die folgenden Aussagen.

1. Eine Menge S ist genau dann in NP, wenn es eine polynomiell beschränkte Relation $R_S \subseteq \Sigma^* \times \Sigma^*$ gibt, so dass R_S in P liegt und $x \in S$ genau dann gilt, wenn es ein y gibt, so dass $(x, y) \in R_S$. Man kann ein solches y als einen Beweis für $x \in S$ betrachten.
2. Für jede polynomiell beschränkte Relation R gilt: Wenn $R' = \{(x, y) \mid \exists y' \text{ mit } (x, yy') \in R\}$ in P liegt, dann ist R in polynomieller Zeit lösbar.
3. Wenn P = NP gilt und S in NP liegt, dann ist R_S in polynomieller Zeit lösbar.

Die dritte Aussage bedeutet, dass unter der Annahme P = NP für jedes Problem $S \in$ NP (z.B. SAT) und jedes $x \in S$ (z.B. eine erfüllbare Formel φ) ein Beweis für das Enthaltensein von x in S (bei SAT also eine erfüllende Belegung der Variablen von φ) effizient berechnet werden kann.

Es folgen die letzten TI2-Hausaufgaben dieses Semesters!

Hausaufgabe 12.4

Die Sprachen L_1 und L_2 seien in $\mathrm{NP} \cap \mathrm{coNP}$. Zeigen Sie, dass dann die Menge $L_1 \oplus L_2 = \{x \mid x$ ist entweder in L_1 oder in $L_2\}$ ebenfalls in $\mathrm{NP} \cap \mathrm{coNP}$ enthalten ist.

Hausaufgabe 12.5

Für eine Sprache L ist eine Orakel-Turingmaschine relativ zu L eine erweiterte Turingmaschine, die in einem Schritt entscheiden kann, ob ein Wort in L liegt. Man definiert P^L als die Menge der Sprachen, die in polynomieller Zeit mit einer deterministischen Orakel-Turingmaschine relativ zu L entschieden werden können. NP^L ist entsprechend über nichtdeterministische Orakel-Turingmaschinen definiert.

Zeigen Sie, dass NP und coNP Teilmengen von $\mathrm{P}^{\mathrm{SAT}}$ sind und dass aus $\mathrm{NP} = \mathrm{P}^{\mathrm{SAT}}$ folgt, dass $\mathrm{NP} = \mathrm{coNP}$ gilt.

Hausaufgabe 12.6

Zeigen Sie, dass eine nichtdeterministische Turingmaschine mit quadratischem Platzaufwand deterministisch simuliert werden kann (Satz von Savitch). Beachten Sie, dass die deterministische TM nicht einfach nacheinander alle möglichen Berechnungszweige der NTM durchprobieren kann. Eine terminierende TM, die bei einer Eingabe der Länge n maximal $f(n)$ Platz verbraucht, kann nämlich $O(c^{f(n)})$ Schritte für ein $c > 1$ bis zur Terminierung benötigen. Weil beim Durchprobieren der möglichen Berechnungszweige der NTM alle bisherigen Entscheidungen gespeichert werden müssten, würde die simulierende deterministische TM exponentiell viel Platz benötigen.

Folgender Ansatz führt zum Ziel: Ob eine NTM N in maximal $t > 1$ Schritten von einer Konfiguration κ_1 zu einer Konfiguration κ_2 gelangen kann, kann dadurch überprüft werden, dass für jede mögliche Konfiguration κ bestimmt wird, ob N in maximal $\lfloor t/2 \rfloor$ Schritten von κ_1 zu κ und in maximal $\lfloor t/2 \rfloor$ Schritten von κ zu κ_2 gelangen kann. Ob N in höchstens einem Schritt von κ_1 zu κ_2 gelangen kann, lässt sich direkt überprüfen.

Geben Sie ein Programm an, dass überprüft, ob eine NTM in maximal t Schritten von einer Konfiguration κ_1 zu einer Konfiguration κ_2 gelangen kann. Analysieren Sie die Platzkomplexität des Programms. Begründen Sie, warum dieses Programm ausreicht, um den Satz von Savitch zu beweisen.