Technische Universität München
Theoretische Informatik
Dr. J. Kreiker / Dr. M. Luttenberger, J. Kretinsky

Summer term 2010
August 2, 2010

SOLUTION

Complexity Theory – Final Exam

***Please note*: If not stated otherwise, all answers have to be justified.**

Last name: ____________________

First name: ____________________

Student ID no.: ____________________

Signature: ____________________

Remarks

- If you feel ill, let us know immediately.
- Fill in all the required information and don't forget to sign.
- Write with a non-erasable pen, do not use red or green color.
- You have 10 minutes to read the questions and, after that, 120 minutes to write your solutions.
- You are not allowed to use auxiliary means other than your pen.
- You may write your answers in English or German.
- Check if you received *8 sheets of paper*.
- Write your solutions directly into the exam booklet. Please ask if you need additional (scrap) paper. Only solutions written on official paper will be corrected.
- You can obtain *40 points*. You need *17 points* to pass *including* potential bonuses awarded.
- Don't fill in the table below.
- Good luck!

Ex1	Ex2	Ex3	Ex4	Ex5	Ex6	Ex7	$\sum$

<u>Exercise 1</u> True/False **each 1P=10P**

Points are awarded as follows:

	unsure ☐	unsure ⊠
correct answer	1P	0.5P
wrong answer	-1P	-0.5P
no answer	0P	0P

Remarks

- All claims are definitely either true or false.
- Tick *unsure* ☐ in case of doubt to reduce a potential point deduction.
- A negative total implies zero points awarded for this exercise.

	true	false	unsure
PP has complete problems.	⊠	☐	☐
$\bigcup_{k\in\mathbb{N}} \mathbf{IP}[n^k] = \bigcup_{k\in\mathbb{N}} \mathbf{AM}[n^k]$.	⊠	☐	☐
If $\mathbf{L} \neq \mathbf{P}$, then **L** is closed under $\leq_p$.	☐	⊠	☐
Every probabilistic TM running in expected polynomial time decides a language in $\mathbf{ZPP} := \mathbf{RP} \cap \mathbf{coRP}$.	⊠	☐	☐
If graph isomorphism is **NP**-complete, then $\mathbf{\Sigma}_2^p = \mathbf{\Pi}_2^p$.	⊠	☐	☐
3SAT has no polynomially-sized, interactive, zero-knowledge proofs.	☐	⊠	☐
For every $L \in \mathbf{IP}$ there is an alternating TM deciding L in polynomial time.	⊠	☐	☐
If every unary language $L \in \mathbf{NP}$ is already in **P**, then $\mathbf{EXP} = \mathbf{NEXP}$.	⊠	☐	☐
If $\mathbf{PH} = \mathbf{PSPACE}$, then $\mathbf{\Sigma}_k^p \subsetneq \mathbf{\Sigma}_{k+1}^p$ for all $k \in \mathbb{N}$.	☐	⊠	☐
$\mathbf{AC} = \mathbf{NC}$.	⊠	☐	☐

Exercise 2 "One-liners" **each 1P=8P**

Give a short answer. For **Claims** state clearly whether they hold or not.

Notation: Let $\mathbf{polyL} := \bigcup_{k\in\mathbb{N}} \mathbf{SPACE}((\log n)^k)$ be the class of languages decidable in poly-logarithmic space.

Claim: $\mathbf{polyL} \subsetneq \mathbf{EXP}$.

Answer: Yes as $\mathbf{polyL} \subsetneq \mathbf{PSPACE}$ by the space hierarchy theorem.

Question: State Ladner's theorem:

Answer: If $\mathbf{P} \neq \mathbf{NP}$, then there exists a $L \in \mathbf{NP} \setminus \mathbf{P}$ which is not **NP**-complete.

Claim: Probably, $\mathbf{NP} \neq \mathbf{PCP}(1/n \cdot \log n, 1)$ as otherwise ...

Answer: $\mathbf{P} = \mathbf{NP}$.

Question: State a **coNL**-complete problem.

Answer: Path

Question: By what kind of certificates is **NL** characterized?

Answer: (poly-length) read-once

Question: We know that unless $\mathbf{P} = \mathbf{NP}$ there is some $\rho \in (0, 1]$ s.t. Max3Sat cannot be ρ-approximated in polynomial time. State a problem for which we know that this holds for every $\rho \in (0, 1]$.

Answer: Indset

Question: The reduction in the proof by Cook-Levin takes a Turing machine M and an input x and maps it on some CNF formula $\phi_{M;x}$. Let $N_{M;x}$ be the number of certificates u s.t. $M(x, u) = 1$. How is $N_{M;x}$ related to $\phi_{M;x}$?

Answer: $N_{M;x}$ is the number of satisfying assignments of $\phi_{M;x}$ (at least when restricting assignments to the variables appearing in $\phi_{M;x}$).

Question:: State a language in $\mathbf{NC}^2 \setminus \mathbf{AC}^0$ other than Parity.

Answer: $\overline{\text{Parity}}$.

Exercise 3 **2P+2P+2P=6P**

Consider the following set $\mathcal{C}$ of complexity classes.

$$\mathcal{C} = \{\mathbf{NC}^2, \mathbf{L}, \mathbf{IP}, \mathbf{ZPP}, \mathbf{PCP}(\log n, 1), \mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p, \mathbf{BPP}, \mathbf{PH}\}$$

(a) Draw a directed graph with nodes $\mathcal{C}$ such that there is a path from A to B if $A \subseteq B$.

(b) The "deterministic space hierarchy theorem" says that for any space-constructible functions f, g with $f(n) \in o(g(n))$ we have

$$\mathbf{SPACE}(f(n)) \subsetneq \mathbf{SPACE}(g(n)).$$

Show that $\mathbf{NL} \subsetneq \mathbf{IP}$.

(c) It is unknown whether $\mathbf{P} = \mathbf{L}$ and whether $\mathbf{P} = \mathbf{PSPACE}$. However, not both equalities can hold at the same time.

Which classes in $\mathcal{C}$ coincide

i) under the assumption that $\mathbf{P} = \mathbf{L}$?

ii) under the assumption that $\mathbf{P} = \mathbf{PSPACE}$?

Solution:

$$\begin{tikzcd}
 & & & \mathbf{BPP} \arrow[dr] & & & \\
\mathbf{L} \arrow[r] & \mathbf{NC}^2 \arrow[r] & \mathbf{ZPP} \arrow[ur] \arrow[dr] & & \mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p \arrow[r] & \mathbf{PH} \arrow[r] & \mathbf{IP} \\
 & & & \mathbf{PCP}(\log n, 1) \arrow[ur] & & &
\end{tikzcd}$$

(a)

(b)
- By Savitch's theorem we have $\mathbf{NL} \subseteq \mathbf{SPACE}((\log n)^2)$.
- By the deterministic space hierarchy theorem, it follows that $\mathbf{SPACE}((\log n)^2) \subsetneq \mathbf{SPACE}(n) \subseteq \mathbf{PSPACE}$.
- As $\mathbf{IP}$ equals $\mathbf{PSPACE}$, the result follows.

(c)
- If $\mathbf{P}$ equals $\mathbf{L}$, then $\mathbf{L}$ equals also $\mathbf{NC}$.
- If $\mathbf{P}$ equals $\mathbf{PSPACE}$, then the remaining six classes coincide.

Exercise 4 **2P+2P+2P+1P=7P**

Given an undirected graph $G = (V, E)$ we call $C \subseteq V$ a *clique* of G if for any two distinct nodes u, v of C there is an edge $(u, v) \in E$. A clique C is *maximal* if $C \cup \{u\}$ is not a clique for any $u \in V \setminus C$.

Consider the following decision and function problems related to cliques.

- $\textsc{Clique}_D := \{\langle G, k\rangle \mid G$ has a clique of size at least $k\}$.
- $\textsc{ExactClique}_D := \{\langle G, k\rangle \mid$ Every maximal clique in G has size exactly $k\}$.
- $\textsc{cliqueSize}(G) := \max\{\ |C| \mid C$ is a clique of $G\}$.

Remark: You may assume that $\textsc{Clique}_D$ is **NP**-complete.

(a) Show how to calculate $\textsc{cliqueSize}(G)$ in time $\mathcal{O}(T(n) \cdot \log n)$ under the assumption that $\textsc{Clique}_D$ can be decided in time $T(n)$ for any graph G of size n.

(b) Recall the definition of **DP**:

$$\mathbf{DP} := \{L_1 \cap L_2 \mid L_1 \in \mathbf{NP}, L_2 \in \mathbf{coNP}\}.$$

Show that $\textsc{ExactClique}_D$ is in **DP**.

(c) You may assume that the following language, $\textsc{ExactIndSet}_D$, is **DP**-complete:

$$\textsc{ExactIndSet}_D := \{\langle G, k\rangle \mid \text{Any independent set of maximal size in } G \text{ has size exactly } k\}.$$

Show that $\textsc{ExactClique}_D$ is **DP**-complete.

(d) Show that for no $\rho \in (0, 1]$ there exists a poly-time ρ-approximation of $\textsc{cliqueSize}$ unless $\mathbf{P} = \mathbf{NP}$.

Solution:

(a) The maximal size of any clique is given by number of nodes of the graph G which is surely bounded by its representation length $n := |G|$. We use binary search to find the optimal value in the interval $[0, n]$: the result of a query $\langle G, k\rangle \in \textsc{Clique}_D$ determines whether to descend into the upper or lower half of the remaining interval. This results in at most $\log n$ calls to $\textsc{Clique}_D$ where every call takes time at most $T(n)$.

(b) *Remark*: There was an error in the definition of $\textsc{ExactClique}$.

D1: The definition should have been

$$\langle G, k\rangle \in \textsc{ExactClique}_D \;:\Leftrightarrow k = \textsc{cliqueSize}(G).$$

similar to $\textsc{ExactIndSet}$.

D2: What we defined instead was:

$$\langle G, k\rangle \in \textsc{ExactClique}_D \;:\Leftrightarrow \forall C : C \text{ is a maximal clique of } G \Rightarrow |C| = k.$$

Still, for both definitions $\textsc{ExactClique}_D$ is in **DP**:

D1: We have

$$\langle G, k\rangle \in \textsc{ExactClique}_D \text{ iff } \langle G, k\rangle \in \textsc{Clique}_D \wedge \langle G, k+1\rangle \notin \textsc{Clique}_D.$$

Define now $L := \{\langle G, k\rangle \mid \langle G, k+1\rangle \in \textsc{Clique}_D\}$. L is clearly in **NP**, so its complement is in **coNP**. We then have

$$\textsc{ExactClique}_D = \textsc{Clique}_D \cap \overline{L} \in \mathbf{DP}$$

Remark: Note that is not sufficient to show that $\textsc{ExactClique}_D$ is in $\mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p$ as we only know that $\mathbf{DP} \subseteq \mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p$.

D2: Obviously, we can decide in polynomial time if a given subset $C \subseteq V$ is a maximal clique and if $|C| = k$. Hence,

$$\forall C\ : C \text{ is a maximal clique of } G \Rightarrow |C| = k$$

is a calculation in $\mathbf{\Pi}_1^p = \mathbf{coNP}$ which is a subset of $\mathbf{DP}$.

Solutions relying on any of the two definitions have been accepted.

(c) *Remark*: This exercise makes only sense w.r.t. definition D1. Every student has therefore been awarded 2P.

Given $G = (V, E)$, set $G' := (V, \{(u, v) \in V \times V \mid u \neq v\} \setminus E)$, i.e., two nodes of G' are connected by an edge iff they are not connected by an edge in G. Now, the cliques of G are in bijection with the independent sets of G'. This gives us the following polynomial time reduction:

$$\langle G, k\rangle \in \textsc{ExactIndeset}_D \text{ iff } f(\langle G, k\rangle) := \langle G', k\rangle \in \textsc{ExactIndset}_D.$$

(d) *Remark*: This exercise makes only sense w.r.t. definition D1. Every student has therefore been awarded 1P.

This follows immediately from the reduction given in (c) and the fact that INDSET is **NP**-hard to ρ-approximate for any $\rho \in (0, 1]$.

Exercise 5 **4P**

A path $v_0 v_1 \dots v_k$ in a directed graph $G = (V, E)$ with nodes V and edges $E \subseteq V \times V$ is a *cycle* if $v_k = v_0$. Define

$$\textsc{Cycle} := \{\langle G, v\rangle \mid \ v \text{ lies on a cycle of } G\ \}.$$

Show that $\textsc{Cycle}$ is $\mathbf{NL}$-complete.

Solution:

- $\textsc{Cycle} \in \mathbf{NL}$:

  v is in some cycle of G iff there is a path from s to s in G, i.e., $\textsc{Cycle}$ is log-reducible to $\textsc{Path}$ via the reduction $f(\langle G, v\rangle) := \langle G, v, v\rangle$. As $\mathbf{NL}$ is closed under $\leq_{\log}$ the result follows.

- $\textsc{Cycle}$ is $\mathbf{NL}$-hard via $\textsc{Path} \leq_{\log} \textsc{Cycle}$:

  Given $\langle G, s, t\rangle$ add a new node c to G which has exactly one outgoing edge (c, s) and one incoming edge (t, c). We can do this in log-space. If there is a path π from s to t in G, then there is now also a cycle $c\pi c$. On the other hand, every cycle ζ which contains c has to start with the edge (c, s) and end with the edge (t, c), i.e., $\zeta = cs\pi tc$ for some path π. We may assume that c does not appear in π by simply taking a cycle ζ of minimal length. Then π is a path from s to t in G.

Exercise 6 **2P**

Explain the error in the following proof of $\mathbf{P} \neq \mathbf{NP}$:

1) Suppose $\mathbf{P} = \mathbf{NP}$.
2) Then for some $k \in \mathbb{N}$, $\textsc{Sat} \in \mathbf{DTIME}(n^k)$.
3) As every language in $\mathbf{NP}$ is reducible to $\textsc{Sat}$, $\mathbf{NP} \subseteq \mathbf{DTIME}(n^k)$.
4) Due to the assumption $\mathbf{P} = \mathbf{NP}$, $\mathbf{P} \subseteq \mathbf{DTIME}(n^k)$.
5) Then $\mathbf{DTIME}(n^k) \subsetneq \mathbf{DTIME}(n^{k+1})$ is a contradiction with $\mathbf{P} \subseteq \mathbf{DTIME}(n^k)$.

Solution: The time needed for calculating the reduction isn't taken into account in step 3.

Exercise 7 **1P+2P=3P**

Consider the following kind of computation:

Starting from a deterministic polynomial-time TM V, we extend V as follows:

- V has two special tapes:
  - a query-tape on which it can write Boolean formulae and
  - a response-tape.
- At any step of the computation, V can send the formula on the query tape to a SAT-prover that immediately tells V if the formula is satisfiable or not by writing either 1 or 0 to the response-tape.

Let $\mathbf{SAT}[k]$ be the class of all languages L which are decided by deterministic polynomial-time TMs asking at most k SAT-questions (where k may depend on the input length).

(a) The Boolean closure of **NP** is the class of all languages L of the form:

$$L := \bigcup_{i=1}^{m} \bigcap_{j=1}^{n} L_{i,j} \text{ where } L_{i,j} \in \mathbf{NP} \cup \mathbf{coNP}$$

Show that for any such language L there exists a constant k s.t. $L \in \mathbf{SAT}[k]$.

(b) Show that $\mathbf{SAT}[\text{poly}(n)] \subseteq \mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p$.

Remark: $\mathbf{SAT}[\text{poly}(n)]$ refers to the class of languages decided by deterministic polynomial-time TMs which can query the SAT-prover a number of times polynomial in the length of the input.

Solution:

(a) For a language $A \subseteq \Sigma^*$ set $A^1 := A$ and $A^{-1} := \Sigma^* \setminus A$. For every language L_{ij} we find an **NP**-language A_{ij} and a $c_{ij} \in \{1, -1\}$ s.t. $L_{ij} = A_{ij}^{c_{ij}}$. Then

$$L = \bigcup_{i=1}^{m} \bigcap_{j=1}^{n} A_{ij}^{c_{ij}}.$$

Now let f_{ij} be the poly-time reduction from A_{ij} to SAT and let $a_{ij}(x)$ be the answer by the SAT-solver for the query $f_{ij}(x) \overset{?}{\in}$ SAT. We then have:

$$x \in L \text{ iff } \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n} a_{ij}^{c_{ij}}(x) = 1 \text{ with } a^1 := a \text{ and } a^{-1} := 1 - a.$$

We therefore need to do at most $n \cdot m$ queries to the SAT-solver and then evaluate the n-CNF formula on the valuation given by the answers by the SAT-solver.

(b) Let M be a det. polynomial-time TM which uses at most a polynomial number of queries to the SAT-prover. Let $q(n)$ be the polynomial bounding the number of queries for an input of length n.

We obtain from M the det. polynomial-time TM N which takes the same input as M and additionally a sequence of Boolean formulae $\phi_1, \ldots, \phi_{q(n)}$ and also a bit string β of length $q(n)$ where $\beta_i = 1$ iff ϕ_i is satisfiable. Of course, all the formulae ϕ_i are of polynomial length (otherwise V wouldn't be able to write them on its query-tape).

On input $\langle x, \phi_1, \ldots, \phi_{q(n)}, \beta \rangle$, N now simulates M on x until the first query. It then checks that the first query of M is indeed the formula ϕ_1 and takes β_1 as the response by the SAT-prover. This way, N simulates M on x without any interaction with the prover. It accepts if M accepts. We now have

$$x \in L \text{ iff } \exists \phi_1, \ldots, \phi_{q(n)}, \beta_1, \ldots, \beta_{q(n)} : N(x, \phi_1, \ldots, \phi_{q(n)}, \beta_1, \ldots, \beta_{q(n)}) = 1 \wedge \bigwedge_{i=1}^{q(n)} \beta_i = 1 \Leftrightarrow \phi_i \in \text{SAT}.$$

Note that

$$\beta_i = 1 \Leftrightarrow \phi_i \in \textsc{Sat} \equiv (\beta_i = 0 \Rightarrow \phi_i \in \overline{\textsc{Sat}}) \wedge (\beta_i = 1 \Rightarrow \phi_i \in \textsc{Sat})$$

is a computation in $\mathbf{NP} \cup \mathbf{coNP}$. The complete computation is therefore in $\mathbf{\Sigma}_2^p$. As **SAT**[k] is closed under complement, we immediately obtain that it is also contained in $\mathbf{\Pi}_2^p$.