Instructor: Rocco A. Servedio

Computer Science 4236: Introduction to Computational Complexity
Problem Set #2 Spring 2023

Due 11:59pm Wednesday, March 1, 2023

See the course Web page for instructions on how to submit homework.
Important: To make life easier for the TAs, **please start each problem on a new page.**

Problem 1 The complexity class DP is defined as follows:

DP = {languages L : there is a language $L_1 \in$ NP and a language $L_2 \in$ coNP such that $L = L_1 \cap L_2$.}

(An equivalent definition is that L is in DP if and only if $L = A \setminus B$ for two languages $A, B \in$ NP, i.e. L is the set-theoretic difference of two languages in NP.)

(a) Define the language ALMOST-SAT to be the language of all CNF formulas which (i) are unsatisfiable, but (ii) would become satisfiable if any clause were removed. Show that ALMOST-SAT is in DP.

(b) Now define the language ONE-SAT to be the set of all Boolean formulas ϕ such that ϕ has *exactly one* satisfying assignment. Show that ONE-SAT is in DP.

(c) Give a characterization of DP in terms of deterministic polynomial-time algorithms equipped with a SAT oracle. Justify your characterization.

As an example of what a characterization of a different complexity class (our friend NP) would look like in terms of polynomial-time algorithms equipped with a SAT oracle, here is such a characterization for NP: a language is in NP if and only if it is decided by a deterministic polynomial-time algorithm that makes one SAT oracle call and accepts if and only if the call accepts.

(The notion of "non-adaptivity" may be useful: an oracle algorithm is *non-adaptive* if it makes all of its calls to the oracle "in parallel" (i.e. all the strings on which the oracle is every queried are selected before the oracle is ever called — the choice of the i-th of those strings doesn't depend on what the oracle answered on the first $i-1$ calls).

Problem 2 Recall the definition of the complexity class DP from Problem 1.

(a) (easy) Let L be the language consisting of all pairs (ϕ, ψ) such that ϕ is a satisfiable Boolean formula and ψ is an unsatisfiable Boolean formula. Show that L is DP-complete.

(b) Here is a fact which can be shown (you don't need to show this): **Fact:** There is a polynomial-time reduction (call the reduction R) from SAT to CLIQUE which has the following property: if the input formula for SAT is satisfiable then the pair (G, k) constructed by R has largest clique of

size exactly k, and if the input formula is not satisfiable then the pair (G, k) constructed by R has largest clique of size exactly $k-1$.

Recall that the LARGEST-CLIQUE language is the set of all pairs (G, k) such that G is an undirected graph and the largest clique in G is of size exactly k.

Show that LARGEST-CLIQUE is DP-complete. (Hint: Use the fact stated above.)

Problem 3 A DNF formula is an OR of "terms", where a "term" is an AND of literals; for example,

$$(x_1 \wedge \overline{x}_3 \wedge x_4) \vee (x_2 \wedge x_3) \vee (\overline{x}_2 \wedge x_4 \wedge x_5 \wedge x_6)$$

is a 3-term DNF. A CNF is an AND of "clauses" where a "clause" is an OR of literals; for example,

$$(x_3 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3)$$

is a 2-clause CNF.

(a) Show that there are DNF formulas with $O(n)$ terms such that any logically equivalent CNF formula must have $2^{\Omega(n)}$ clauses.

(b) Consider the following decision problem: The input is a pair $(\phi, 1^k)$ where ϕ is a DNF formula and k is an integer. The problem is to determine whether there is a CNF formula that is logically equivalent to ϕ that has k or fewer clauses.

Show that this problem (or more precisely, the language corresponding to this problem) is in Σ_2^p.

Problem 4 We say that an algorithm A makes *non-adaptive* queries to an oracle if it selects all of the strings on which it will query the oracle before querying the oracle on any of those strings (in other words, the queries to the oracle can be thought of as being carried out "in parallel" — the choice of later query strings can't depend on the answers that the oracle provides to earlier queries). In contrast, an *adaptive* oracle algorithm makes its queries sequentially (in particular, the choice of the i-th query string for the oracle can depend on the answers that the oracle gave to query strings $1, \ldots, i-1$).

Show that a language L is decided by a deterministic polynomial-time algorithm that has non-adaptive access to an NP oracle, if and only if L is decided by a deterministic polynomial-time algorithm that has adaptive access to an NP oracle but can make only $O(\log n)$ queries.

Problem 5

(a) Let's say that a polynomial-time reduction R from language A to language B is a *shrinky-dink reduction* if it has the following property: for every $x \in \Sigma^\star$, the output $R(x)$ of reduction R on input x is a string of length $|R(x)| \leq O(\log |x|)$.

Show that if there is a polynomial-time shrinky-dink reduction from CLIQUE to L where L is a language in NP, then NP $\subseteq \cup_{k \geq 1} TIME(n^{(\log n)^k})$. (The latter complexity class is sometimes referred to as "quasipolynomial time.")

(b) Let $S(n) \geq \log n$ be a proper complexity function, and let L be a language which is *accepted* by some $S(n)$-space bounded Turing machine M. In other words, M never uses more than $S(n)$ space on any input of length n, and M halts and accepts on precisely those inputs x which belong to L; however, for inputs $x \notin L$, M may either halt and reject or may run forever.

Show that there is a $S(n)$-space bounded Turing machine M' that *decides* L. In other words, M' halts on every input string x and accepts precisely those $x \in L$, and never uses more than $S(n)$ space on any input of length n.

Problem 6 This problem deals with the "inclusion translates upward, separation translates downward" phenomenon we discussed in class.

(a) Let $f(n) \geq 10n$ be a proper complexity function. Show that if $TIME(n) = TIME(n(\log n)^{2/3})$ then $TIME(f(n)) = TIME(f(n)(\log f(n))^{2/3})$.

(b) Show that $TIME(n) \neq TIME(n(\log n)^{2/3})$. (Hint: use part (a) and a hierarchy theorem.) Note that this is a finer separation than could be achieved just by directly applying the time hierarchy theorem.