Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

Homework 5
Due: 10:00am, Tuesday October 10

---

1. **(Courtroom complexity.)** In this problem we study a slightly peculiar complexity class that we'll call $\mathsf{S_2P}$. Informally, we say $L \in \mathsf{S_2P}$ whenever the following circumstances hold. There are two lawyers, Yolanda and Zeyuan, whose job is to argue in front of judge Victor about whether or not $x \in L$. Whenever $x \in L$, there is something Yolanda can say that will convince judge Victor that indeed $x \in L$, no matter what Zeyuan says. Conversely, whenever $x \notin L$, there is something Zeyuan can say that will convince judge Victor that $x \notin L$, no matter what Yolanda says.

   More precisely, we say that $L \in \mathsf{S_2P}$ if there is a polynomial $p(n)$ and a polynomial-time algorithm V such that

   $$x \in L \implies \exists^p y\ \forall^p z\ V(x,y,z) = 1,$$
   $$x \notin L \implies \exists^p z\ \forall^p y\ V(x,y,z) = 0.$$

   (Recall "$\exists^p y$" means "$\exists y$ with $|y| \leq p(|x|)$", etc.)

   (a) Show that $\mathsf{S_2P}$ is closed under complement: $\mathsf{coS_2P} = \mathsf{S_2P}$.

   (b) Show that $\mathsf{S_2P} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$.

   (c) Show $\mathsf{NP} \subseteq \mathsf{P/poly} \implies \mathsf{PH} = \mathsf{S_2P}$. (This is an improvement on the Karp–Lipton Theorem, by part (b)... but in fact, you can solve this problem by almost literally repeating the proof of Karp–Lipton.)

   (d) Show that $\mathsf{P^{NP}} \subseteq \mathsf{S_2P}$.

2. **(A route to $\mathsf{P} \neq \mathsf{NP}$?)** Let c_n denote the maximum number of gates needed by a Boolean circuit to compute any function $f : \{0,1\}^n \to \{0,1\}$. Shannon and Lupanov showed that $c_n \approx 2^n/n$, but we will be interested in the literal exact value of c_n. Let us say that a language L has *maximal circuit complexity* if $L \cap \{0,1\}^n$ requires circuits of size c_n for every n.

   Show that if every language in E has *non*-maximal circuit complexity (i.e., just *one* gate can be saved somewhere in the circuit family) then $\mathsf{P} \neq \mathsf{NP}$. (Recall that $\mathsf{E} = \bigcup_c \mathsf{TIME}(2^{cn})$.)

3. **(Limited SAT queries.)** When C is a complexity class, the notation $\mathsf{C}^{A[k]}$ means the same class where *at most* k oracle queries to the language A are allowed. As usual, $\mathsf{C}^{\mathsf{NP}[k]}$ denotes the union of $\mathsf{C}^{A[k]}$ over all $A \in \mathsf{NP}$; equivalently, it's $\mathsf{C}^{\mathrm{SAT}[k]}$. In studying the Polynomial Time Hierarchy, we observed that when $\mathsf{C} = \mathsf{NP}$, we could massively reduce the number of queries used: $\mathsf{NP^{NP}} = \mathsf{NP}^{\mathsf{NP}[\mathrm{poly}(n)]} = \mathsf{NP}^{\mathsf{NP}[1]}$. The same is (seemingly) not true when $\mathsf{C} = \mathsf{P}$; it is believed that $\mathsf{P}^{\mathsf{NP}[1]} \subsetneq \mathsf{P}^{\mathsf{NP}[2]} \subsetneq \mathsf{P}^{\mathsf{NP}[3]} \subsetneq \cdots$

   In this problem, we will look at an interesting class: $\mathsf{P}^{\mathsf{NP}[\log]}$, which is short for $\mathsf{P}^{\mathsf{NP}[O(\log n)]}$, the class of languages decidable in polynomial time by a SAT-oracle Turing Machine that makes at most $O(\log n)$ oracle queries on inputs of length n.

   (a) Show that the following two problems are in $\mathsf{P}^{\mathsf{NP}[\log]}$: UNIQUE-MAX-CLIQUE, the language of all graphs whose largest clique is unique; ODD-MAX-CNF-SAT, the language of all CNF formulas for which the maximum number of clauses that can be satisfied by any truth assignment is odd.

(b) Define $\mathsf{P}_{\|}^{\mathsf{NP}[r]}$ to be the class of all languages decidable in polynomial time by a SAT-oracle Turing Machine that makes at most r *nonadaptive* oracle queries. This means that the machine can only interact with "the oracle" one time, in the following way: it can submit r separate oracle queries, and get back the r answers. Show that $\mathsf{P}^{\mathsf{NP}[k]} \subseteq \mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]}$, even for $k = O(\log n)$, and hence $\mathsf{P}^{\mathsf{NP}[\log]} \subseteq \mathsf{P}_{\|}^{\mathsf{NP}}$.

(c) Building on work of Gilbert, Michael Fischer showed the following result: For every n, there is an n-input, n-output Boolean circuit, consisting of poly(n) AND gates, poly(n) OR gates, and $\lceil \log_2(n+1) \rceil$ NOT gates, such that on input $(x_1, x_2, \ldots, x_n)$, the output is $(\neg x_1, \neg x_2 \ldots, \neg x_n)$.[1] If you have never seen this before, I very strongly urge you to try to prove this result in the case $n = 3$; it's a great puzzle! But anyway, you can assume Fischer's result.

Show an almost-opposite containment to part (b): $\mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]} \subseteq \mathsf{P}^{\mathsf{NP}[k+1]}$, even for $k = O(\log n)$, and hence $\mathsf{P}^{\mathsf{NP}[\log]} = \mathsf{P}_{\|}^{\mathsf{NP}}$.

(0-point bonus problem: Can you get the exact-opposite containment, $\mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]} \subseteq \mathsf{P}^{\mathsf{NP}[k]}$ in case $k = 2$? Can you get it in general?)

[1] Also, the construction is P-uniform.