CSE696, Spring 2021 **Problem Set 3** **Due Mon. 4/26**

Reading:

We are transiting from Arora-Barak chapter 10, (plus sections 16.1–16.3 of chapter 16) to quantum computing with emphasis on Simon's, Shor's, and Grover's algorithms. Note that the Arora-Barak text has its own chapter 20 on quantum computing which focuses on these topics.

(1) A three-part progression of worst-case one-way function notions. In each case, f is stratified so that $m = |f(x)|$ depends only on $n = |x|$, so we can write $f = [f_n]$ as a family of finite functions if we wish. And we suppose the time $t(n)$ to compute $f(x)$ for all x of length n is polynomial in n. In each case, we define

$$I_f = \{\langle w, y\rangle : (\exists x) : w \sqsubseteq x \wedge f(x) = y\}.$$

This is a language version of the problem of inverting f, where $w \sqsubseteq x$ means that w is a prefix of x, i.e., there exists a string v such that $x = wv$. One could alternatively use $w \leq x$ as the condition instead without changing the nature of this problem.

(a) Suppose f is a permutation of $\{0,1\}^*$ that permutes $\{0,1\}^n$ for each n. Show that $I_f \in \mathsf{UP} \cap \mathsf{co\text{-}UP}$.

(b) Now suppose $f = [f_n]$ is such that each f_n maps $\{0,1\}^n$ 2-to-1 into $\{0,1\}^n$, with half of $\{0,1\}^n$ being not in the range of f_n. Show that $I_f \in \mathsf{UP}$. Explore the question of whether $I_f \in \mathsf{co\text{-}UP}$ a little and say what the obstacle(s) are.

(c) Now remove the 2-to-1 condition. Give an example of a family $f = [f_n]$ with $m(n) = \tilde{\Theta}(n)$ such that I_f is NP-complete. Can you arrange $m(n) = n$ exactly? (Hint: Think of building up prefixes of satisfying assignments that f erases. Maybe include some padding. $6 + 9 + 15 = 30$ pts.)

(2) Suppose $\mathsf{UP} \neq \mathsf{P}$. Produce a family $f = [f_n]$ of the 2-to-1 kind in 1(b) that cannot be inverted in polynomial time. (12 pts.)

(3) The logic of using $a = \frac{1}{3}2^k$ and $b = \frac{2}{3}2^k$ as the hypothesized interval for $|S|$ given k is that as you go from 2^k to 2^{k-1} or 2^{k+1}, these intervals neatly knit together to cover all possible nonzero values of $|S|$ from 1 to 2^n. They give n intervals, so the probability $\geq \frac{2}{9}$ which my notes obtain (corresponding to $\frac{1}{8}$ in Arora-Barak, which becomes overall $\frac{1}{8n}$) for each interval translates into the overall success probability $\geq \frac{2}{9n}$.

What happens if we use wider or narrower intervals $[a, b]$? For instance, what if we step from k to $k-2$ and $k+2$, so that we only have $\frac{n}{2}$ intervals, and make $a = b/4$ so that they still tile? Does this make the lower bound on the overall success probability better than $\frac{2}{9n}$? What is the limit that seems to be possible with this approach? (12 pts. for the $a = b/4$ calculation and at least 6 pts. for exploring the limit question.)

(4) (An alternate proof of the first part of Toda's Theorem that uses odd-parity more generally than the fact of 1 being an odd number): Let $K = 2^{q(n)}$ and $N = 2^{r(n)}$ where $r(n)$ is the number of random bits the $\mathsf{BP} \cdot \oplus\mathsf{P}$ machines we are building will be allowed. Say that a $K \times N$ matrix G with 0-1 entries is *good* if:

- Any given entry $G[i, j]$ can be computed in time polynomial in $q(n) + r(n)$—note that this is the length of i as a $q(n)$-bit number plus that of j as an $r(n)$-bit number.
- For every i, $1 \leq i \leq K$, row i has at least $N/8$ 1's. Moreover, so does every N-vector obtained by XOR-ing any subset S of the rows of G.

Take for granted that there exist families $[G_n]$ of good matrices for any polynomials $q(n)$ and $r(n)$, which by the first condition gives polynomial time in n overall. Indeed, they can be built with $G_n[i, j]$ computable in time $(q(n) + r(n))$ times a polynomial in $\log n$. Use this to show $\mathsf{NP} \subseteq \mathsf{RP}[\oplus\mathsf{P}]$. Compare the efficiency of the reduction in terms of $q(n)$ and $r(n)$ and the overall "success of oddness" probability with $\frac{2}{9n}$ or etc. in problem (3). (30 pts., for 90 to this point, before one more problem to come).

(5) Take $[N_i]$ to be a fixed and natural recursive presentation of polynomial-time bounded NTMs, each with its associated polynomial time bound p_i (which you may lavishly or slavishly take to be $n^i + i$). Take $\mathcal{F}$ to be any strong, sound, and effective system of logic, with proof predicate $P_{\mathcal{F}}$ as on Assignment 2. Choose $\mathcal{C}$ to be any one of the following "promise classes": UP, RP, $\mathsf{NP} \cap \mathsf{co\text{-}NP}$, BPP, or (looking ahead) BQP. Without caring about its details, you may take $S_{\mathcal{C}}(i)$ to be a predicate defining "N_i represents a language in $\mathcal{C}$." Two notes:

- For BPP, the language represented by N_i won't be literally $L(N_i)$; most often $L(N_i) = \Sigma^*$ while you want the language of inputs x having over 3/4 witnesses y, for instance.
- For $\mathcal{C} = \mathsf{NP} \cap \mathsf{co\text{-}NP}$ you should use $S_{\mathcal{C}}(i) \equiv (\exists j)[L(N_i) =\sim L(N_j)]$ so that you can reference an N_j in your set B below.

Then define $\Pr\mathcal{C}$ to be the set of languages represented by N_i such that $(\exists d)P_{\mathcal{F}}(S_{\mathcal{C}}(i), d)$. Technically the extent of "provable $\mathcal{C}$" depends on the system $\mathcal{F}$ but as long as $\mathcal{F}$ is natural and reasonable, the features of this problem are all the same.

(a) Construct a language B that is factually in $\mathcal{C}$, such that for all languages $A \in \Pr\mathcal{C}$, $A \leq_m^p B$. In fact, make B be in linear or quasi-linear time in some appropriate sense. (If you choose $\mathcal{C} = \mathsf{NP} \cap \mathsf{co\text{-}NP}$, be sure to show that both B and $\sim B$ belong to NP.)

(b) Address the question of whether $B \in \Pr\mathcal{C}$, so that it would become a complete language for the provable part of $\mathcal{C}$. Why doesn't it simply follow? After all, your answer to part (a) was a proof that B belongs to $\mathcal{C}$. (This is open-ended. I've often wondered whether there is a "Rosser-type trick" that would do an end-run around Gödel here. 24 pts. for (a) and 6+ points for (b), making 120+ total on the set.)

(*) Bonus Question(?) With reference to the 4/17 lecture, consider Boolean circuits C with m inputs that try to predict the next bit of a pseudorandom generator $g(x)$ that outputs m bits in total. When $k < m$, we suppose that the remaining $m - k$ input gates are given a special symbol # that makes any gate involving it return `false`.

(a) Prove an analogue of the equivalence of the Yao and Blum-Micali definitions of security with negligible error against $n^{O(1)}$-sized circuits. (The first part was mostly done in lecture. I think it works the other way...hmm...)

(b) What happens if we fill in the remaining $m-k$ gates by 0s instead? The thing to note is that if $g(x)_k = 0$, then C will automatically make the same prediction for $g(x)_{k+1}$ that it made for $g(x)_k$. Does this dependence destroy the probabilistic reasoning (which is already dented by the absence of "ρ")?