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

Homework 3
Due: 10:00am, Tuesday September 26

---

1. Define $\mathsf{P}/\log$ (polynomial time with logarithmic advice) to be $\bigcup_{b,c} \mathsf{TIME}(n^b)/c \log n$. Show that $\mathsf{NP} \subseteq \mathsf{P}/\log \implies \mathsf{NP} = \mathsf{P}$. (Hint: use the fact that SAT is NP-complete.)

2. Exercise 6.17 in Arora–Barak asks you to prove their Theorem 6.32. We'll get to that result later in the class, but for now, prove the following variant:

   $L \in \mathsf{NP}$ iff L can be computed by a "DC-uniform"[1] circuit family (C_n) with the same four conditions as in Theorem 6.32, except condition #3 is changed to "semi-unbounded fan-in", which means the OR gates can have arbitrary (exponential) fan-in, but the AND gates can only have poly(n) fan-in.

3. A *restarting* probabilistic Turing Machine is a standard probabilistic Turing Machine (Definition 7.1 in Arora–Barak) with the following additional feature: It has a special state called Restart, and if ever the TM transitions into that state, the entire computation is completely restarted (tapes/heads/state all reset to their initial values, nothing remembered from the prior computation). Each computation path from the initial state to Accept/Reject/Restart is called a *run*, and depending on the machine's "coin flips", it may well have multiple runs (restarts) before it finally accepts/rejects. The running time $t(n)$ on inputs of length n is defined to be the maximum possible number of steps in a single run (over all inputs and coin flips). Note that we do *not* sum the times over all runs; we only "pay for" the longest run. (This makes restarting TMs a rather non-realistic model.) Finally, if a restarting TM has the property that it restarts with probability 1 on at least one input, we deem it to be an *invalid* machine.[2]

   Let $\mathcal{C}$ be a probabilistic complexity class such as RP, BPP, PP (or even NP) that can be defined as follows, for some constants $0 \le s \le c < 1$: "$L \in \mathcal{C}$ if there exists a polynomial-time probabilistic TM M such that $x \in L \implies \mathbf{Pr}[M(x) \text{ accepts}] > c$ and $x \notin L \implies \mathbf{Pr}[M(x) \text{ accepts}] \le s$". We then define $\mathsf{Restarting}\mathcal{C}$ to be the same class, except that M is allowed to be a (valid) restarting probabilistic TM.

   (a) Prove that $\mathsf{RestartingPP} = \mathsf{PP}$.

   (b) Prove that $\mathsf{RestartingNP} = \mathsf{NP}$.

   (c) Prove that $\mathsf{RestartingRP} = \mathsf{NP}$.

   (d) $\mathsf{RestartingBPP}$ is a funny class. Prove that $\mathsf{NP}, \mathsf{coNP} \subseteq \mathsf{RestartingBPP}$.

   (e) Prove that $\mathsf{RestartingZPP} = \mathsf{NP} \cap \mathsf{coNP}$, where $\mathsf{RestartingZPP}$ means the class of languages L for which there is a polynomial-time restarting TM with the following properties: Besides "Accept" and "Reject", it has a third halting (output) state called "?". And on input x, the machine (eventually, after possible restarts) outputs "?" with probability at most 1/2, and otherwise outputs the correct answer to $x \overset{?}{\in} L$.

---

[1]This is basically the same as "DLOGTIME-uniform" as discussed in class, except DC-uniformity allows you polylog time, as opposed to logarithmic time. Go with "DC-uniformity" here (as in Arora–Barak's Definition 6.31) for simplicity. However, please augment their definition as follows: not only should the "TYPE" function give a gate's type, it should also give the *number* of incoming wires (i.e., the fan-in amount).

[2]This is nothing unusual; it's exactly like how deterministic TMs may fail to halt on some inputs, in which case we deem them "non-deciders" and say they don't compute any language.