CS 109
Spring 2011
Theory of Computation: Advanced

Homework 7
Due Mon May 2, 5:00pm

Prof. Amit Chakrabarti
Computer Science Department
Dartmouth College

General Instructions: Same as in Homework 1.

Honor Principle: Same as in Homework 1. Additionally, note that the first problem is a standard exercises in a number of textbooks, sometimes with solutions given. You are specifically required **not to consult any books or websites** other than this course's textbooks and website while working on those problems.

14. An *unbounded fan-in circuit* is just like the circuits we defined in class — i.e., DAGs whose vertices (gates) are inputs $(x_1, \ldots, x_n)$, negated inputs $(\neg x_1, \ldots, \neg x_n)$, and logic gates (AND/OR), with one or more gates of fan-out 0 designated as output(s) — except that the restriction that gates have fan-in 2 is removed. Size and depth are defined as before: number of edges (wires) and maximum path length, respectively.

    (Warm-up exercise; no credit; no need to turn this in) Prove that there exists a suitable size function $s : \mathbb{N} \to \mathbb{N}$ such that every Boolean function $f : \{0,1\}^n \to \{0,1\}$ has unbounded fan-in circuits with $O(1)$ depth and $O(s(n))$ size. Find the smallest possible $s(n)$ you can.

    For integers $i \geq 0$, the complexity classes AC^i are defined as follows:

$$\mathsf{AC}^i = \{L \subseteq \{0,1\}^* : \exists c \in \mathbb{N}(L \text{ is decided by an unbounded fan-in circuit family } \langle C_n \rangle_{n=1}^{\infty} \text{ with } \operatorname{size}(C_n) = O(n^c) \text{ and } \operatorname{depth}(C_n) = O(\log^i n))\}\,.$$

    In other words, AC^i is the unbounded fan-in analogue of NC^i. Note that AC^0 does not suffer from the severe shortcoming that NC^0 does, where the output can only depend on a constant number of inputs.

    Prove that $\bigcup_{i=0}^{\infty} \mathsf{AC}^i = \bigcup_{i=0}^{\infty} \mathsf{NC}^i$. [2 points]

15. Let $k > 0$ be an integer. Construct a language in PH that is not in $\mathsf{SIZE}(n^k)$. [2 points]

    This is a hard problem. To get started, recall from the lectures that we proved that there *exist* languages in $\mathsf{SIZE}(n^{2k})$ that are not in $\mathsf{SIZE}(n^k)$. Try writing out this fact formally, using quantifiers: you should have a small, fixed number of quantifier alternations. This suggests that you might be able to place the required language in either Σ_i^p or Π_i^p, for some *fixed* i, independent of k.

    A further hint is given on the next page. I *strongly* recommend that you turn the problem over in your mind for a day at least, before looking at that hint.

16. Give a full formal proof that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$. [2 points]

Homework 7
Due Mon May 2, 5:00pm

Hint for Problem 15: Did you think about the problem for a day, at least? If not, please do!

For a string $w \in \{0,1\}^n$, let B_w denote the Boolean circuit described by w; if w is not a well-formed encoding of a circuit due to syntax errors, define B_w to be a trivial circuit that always outputs 0 (say). Let $C(x)$ denote the output of circuit C on input x. Argue that, for $s \in \mathbb{N}$ and $w, x \in \{0,1\}^*$, the predicates "size$(B_w) \le s$" and "$B_w(x) = 1$" are decidable in polynomial time. Therefore, if we use a fixed number of quantifier alternations and then perform an inner computation that involves evaluating these types of predicates (maybe a few times), we'll have either a Σ_i^p or a Π_i^p computation, depending upon whether we start with a "$\exists$" or a "$\forall$" quantifier.

Now consider the following statement $\phi_n(x)$, for an $x \in \{0,1\}^n$, and figure out what it's saying:

$$\phi_n(x) \;=\; \exists\, w \in \{0,1\}^* \;(\text{size}(B_w) \le n^{2k} \;\wedge\; B_w(x) \;\wedge\; \forall\, v \in \{0,1\}^* \;(\text{size}(B_v) \le n^k \Rightarrow \exists\, y \in \{0,1\}^n \;(B_v(y) \ne B_w(y)))) \,.$$

Once you have digested it, you'll find that this is close to what we need to create a suitable language in PH. But unfortunately $\{x \in \{0,1\}^* : \phi_{|x|}(x)\}$ ends up contaning *all* sufficiently long Boolean strings, so this is not the language we seek! Figure out why this happens, and then think of what you can do to fix it. Perhaps you need more quantifiers.