Midterm
Computational Complexity

February 28, 2014

You are not allowed to collaborate or refer to any book or online source (except possibly to look at the definitions). Due on Monday, March 26 in the class.

1. A language L is called *unary* if $L \subseteq 1^*$. Show that if a unary language is NP-complete, then P=NP.

   Hint: If there is a polytime reduction from SAT to a unary language, then this reduction will map a boolean formula of size n to a string 1^i with $i \leq poly(n)$. Use this and downward self-reducibility of SAT to efficiently find a satisfying assignment. Start out with a recursive brute-force (i.e. depth-first-pruning) algorithm for finding a satisfying assignment. Modify it so that the algorithm runs in polytime.

2. For any positive integer k, show that there is a language in PH with circuit-complexity $\Omega(n^k)$. In fact you can exhibit such a language in some fixed finite level of PH, say Σ_5.

3. Let BPL be the class of languages accepted by a polytime randomized logspace machine with two sided error. Show that BPL $\subseteq$ P.

   Hint: Construct the configuration graph for a BPL *machine, with two outgoing edges from each node, representing the action taken by the machine depending upon the random bit read. The acceptance probability is then the probability that a random walk from the start vertex ends in the accept vertex. How do you calculate this probability efficiently?*

   Note: A randomized logspace machine has workspace of size $O(\log(n))$ and access to $poly(n)$ random bits. The random bits are available on a read-once random tape. This means that once the algorithm reads *a random bit, it cannot go back and read the same bit again from the random tape. A language L is accepted by such a machine if strings in L are accpetd with probability at least* 0.9 *and strings not in L are accpeted with probability at most* 0.1.