CSE681, Spring 2012 **Problem Set 2** **Due Fri. 3/2**

Reading and Exercises:

From the Allender-Loui-Regan notes, please begin the section 29.4 on representing languages via formal systems of logic. This material, in tandem with lecture A01, will go more slowly.

(1) Take the AC^1 circuits for GAP (which is also called *s-t connectivity* from the Friday 2/17 lecture, which use a full binary tree of Boolean matrix multiplications. Each multiplication has n^2 unbounded fan-in OR gates, with each OR gate connected to n binary AND gates. Thus the "only" difference from the NC^1 simulation in previous lectures is that the OR gates have unbounded fan-in. Recall that the NC^1 simulation gives an algorithm that simultaneously runs in polynomial time and $O(\log n)$ space, which puts NC^1 inside the first level SC^1 of the "Steve's Class" hierarchy. Here SC^k means problems solvable by algorithms that run simultaneously in $n^{O(1)}$ time and $O(\log^k n)$ space, and $\mathsf{SC} = \cup_k \mathsf{SC}^k$.

(a) Show what happens when you try to imitate the NC^1 algorithm. Note that as you evaluate an unbounded fan-in OR gate, if you get a 1 returned by the recursive evaluation of any of its inputs, you can immediately return a 1 value and pop back to the parent gate of the OR that you were previously evaluating. (Note that here we are even letting you assume that the log-depth circuit is a tree!—oddly having the root at the bottom as an output gate is often called being "upside down" because it makes the parent relation point downward.) Thus you don't need to store the values returned by the gates fanning in to the OR gate, just remember that the gate hasn't been finished yet. So why can't you do it? Give a detailed explanation—as parts (b) and (c) also guide you to do.

(b) Is the tree of Boolean matrix multiplications really a tree when viewed as an AC^1 circuit? Suppose you insist in using $O(\log n)$ space. Can you do the algorithm at all?

(c) OK, maybe we do need to use $O(\log^2 n)$ space for the addressing of gates, even though it's terribly redundant. But why aren't we getting polynomial time? Are we lazy? Stupid? Show the running time that you get.

(d) Now what if we allow relaxing the space to any power of $\log n$. Getting polynomial time under those conditions would show $\mathsf{NL} \subseteq \mathsf{SC}$, which would be a major breakthrough result though no one has posted a $1,000,000 reward for it.

Points are 12+9+9+6 with especially part (d) being of an open-ended possible extra-credit nature.

(2) Show that the language of undirected graphs that contain a triangle belongs to FO. You may use either an adjacency-matrix or edge-list representation for the graph. Some of you have already done this, so this part is only 6 pts.—note for future reference that FO automatically implies that you get $\mathsf{DLOGTIME}$-uniform circuits.

But for the real question, show that there exists an algorithm for solving this problem that runs in time $O(n^{2.373})$, which is really time $O(N^{1.1865})$ since $N = n^2$ is the true input size. You are allowed to Google the number 2.373 as a hint, and you need not write out the algorithm in full (this would take dozens of pages), just explain why it works on this particular problem. (18 pts., for 24 total on this problem).