Qualifying Examination
Theoretical Computer Science
Friday, March 2, 2012

Part II: Automata and Complexity

Instructions:

1. This is a closed book exam.
2. The exam has four problems worth 25 points each. Read all the problems carefully to see the order in which you want to tackle them. You have all day (9am–5pm) to solve the problems.
3. Write clearly and concisely. You may appeal to some standard algorithms/facts from text books unless the problem explicitly asks for a proof of that fact or the details of that algorithm.
4. If you cannot solve a problem, to get partial credit write down your main idea/approach in a clear and concise way. For example you can obtain a solution assuming a clearly stated lemma that you believe should be true but cannot prove during the exam. However, please do not write down a laundry list of half baked ideas just to get partial credit.

May the force be with you.

Problem 1:

1. Prove that if L is an infinite recursively enumerable language, then there exists an infinite subset R of L which is recursive.
2. Prove that if R is an infinite recursive language, then there exists an infinite subset L of R which is recursively enumerable but not recursive.

Problem 2: Let M_i denote the ith Turing machine. Recall that a many-one reduction from L_1 to L_2 is a computable function f such that for every x, $x \in L_1$ iff $f(x) \in L_2$; we will denote that by $L_1 \leq_m L_2$. A language L is r.e.-hard iff every r.e. language L' reduces to L.

A language L is said to be *productive* iff there is a computable function g such that for every i, if $L(M_i) \subseteq L$ then $g(i) \in L \setminus L(M_i)$; in other words, productive languages are "effectively" non-r.e.

1. If $L_1 \leq_m L_2$ and L_1 is productive, prove that L_2 is productive.
2. Prove that if L is r.e.-hard then $\overline{L}$ is productive. *Hint:* Show that $L_d = \{i \,|\, i \notin L(M_i)\}$ is productive and use the previous part.

Problem 3: Suppose you are given an oracle which can compute the permanent of a matrix on $1-\delta$ fraction of all possible matrices in $\mathbb{F}^{n\times n}$, where $\mathbb{F}$ is a finite field with $|\mathbb{F}| \geq n+2$, and $\delta < \frac{1}{3n}$. Using this oracle, give a polynomial time algorithm to compute the permanent of *any* matrix in $\mathbb{F}^{n\times n}$ correctly with high probability (say $1-2^{-n}$).
[*Hint: Given a matrix A, for a random matrix R, consider the polynomial* $f(x) = \mathrm{p}erm(A+xR)$. *Your goal is to evaluate* $f(0)$.]

Problem 4: Recall the "certificate-based" definition of **NL**: $L \in$ **NL** iff there is a deterministic log-space Turing Machine M, with an additional *read-once* certificate-tape (i.e., the head can move in only one direction) such that

$$L = \{x : \exists w, |w| = \mathrm{poly}(|x|), M \text{ accepts input } x \text{ when given certificate } w\}.$$

Show that if the definition is altered by removing the read-once restriction from the certificate tape, then the resulting class is exactly **NP**.