Qualifying Examination
Theoretical Computer Science
Tuesday, March 6, 2018
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: For a string $w \in \{0,1\}$, $\mathbf{num}(w)$ is the number whose binary representation is w, i.e.,

$$\mathbf{num}(w) = \begin{cases} 0 & \text{if } w = \epsilon \\ 2\mathbf{num}(u) + a & \text{if } w = ua \text{ where } u \in \{0,1\}^*,\ a \in \{0,1\}. \end{cases}$$

For $L \subseteq \{0\}^*$, define $\mathbf{bin}(L) = \{w \in \{0,1\}^* \mid 0^{\mathbf{num}(w)} \in L\}$.

(a) Prove that if $L \subseteq \{0\}^*$ is regular, then $\mathbf{bin}(L)$ is regular.

(b) Give an example of a non-regular language $L \subseteq \{0\}^*$ such that $\mathbf{bin}(L)$ is regular.

Problem 2: A function $f : \Sigma^* \to \Sigma^*$ is *polynomially honest* if there is a polynomial p such that $|w| \leq p(|f(w)|)$. A function f is *one-way* if it is polynomially honest, 1-to-1, and is computable in polynomial time, such that any function g satisfying $g(f(w)) = w$ for all w is not computable in polynomial time. (Note that the definition of one-way functions here is not the standard definition of one-way functions in cryptography, because in cryptography the non-invertability result must hold for most inputs, and the function doesn't have to be 1-to-1.)

UP is the class of problems that are solved in **NP** by machines that have unique accepting computation. In other words, $L \in \mathbf{UP}$ if there is nondeterministic Turing machine M running in polynomial time such that if $x \in L$ then M has a unique accepting computation on x, and if $x \notin L$ then M has no accepting computation on x.

(a) Prove that there is a one-way function iff $\mathbf{P} \neq \mathbf{UP}$.

(b) Prove that there is a one-way function whose range is in **P** iff $\mathbf{P} \neq \mathbf{UP} \cap \mathbf{co\text{-}UP}$.

Problem 3: Consider the following problem: given an undirected graph G, count the number of perfect matchings in G. Suppose there is a polynomial-time algorithm that can approximate the number of perfect matchings to within a factor of n^c for any graph G with n vertices, where c is a constant. Prove that we can approximate the number of perfect matchings to within a $(1 + \varepsilon)$-factor for any fixed $\varepsilon > 0$ in time that is polynomial in n and $1/\varepsilon$.

Problem 4: Recall the Turing-Machine model of space-$s(n)$ computation for general $s(n)$, where the input is on a read-only tape but there is an additional read/write work-tape of which only $s(n)$ cells are ever used for any input of size n. For the randomized variant of this model, the machine has two transition functions, each of which happen with probability $1/2$ at each step.

In this setup, a language L is in **BPL** (Bounded-Error Probabilistic Logspace) if for every string x, and for every possible application of the transition functions, the machine halts and uses at most $O(\log n)$ space. Further, if $x \in L$ then the machine accepts x with probability $\geq 2/3$, and if $x \notin L$ then the machine accepts x with probability $\leq 1/3$.

(a) Show that for any string x of length n, the number of configurations of the machine when computing on x is at most poly(n).

(b) Show that the probability the **BPL** machine accepts x can be computed exactly in polynomial time. Pay attention to the size of the numbers. Hint: dynamic programming.

(c) Conclude that $\mathbf{BPL} \subseteq \mathbf{P}$.