CS 262: Computational Complexity *Instructor: Sandy Irani*

Midterm Exam

Due: May 16, 2018

Important note: Please remember that you should not discuss this exam with anone else or search the internet to look for solutions.

1. Show that if **NP** $\subseteq$ **BPP** then **RP** = **NP**.

   Assume an encoding of SAT in which inputs can be a Boolean input variable x_i or the value 0 or 1. We will assume an encoding of SAT in which every Boolean formula with the same number of inputs (x, 0, or 1) has the same size. This allows us to set an input variable in $\phi(x_1, \ldots, x_n)$ without changing the size of the instance. So $\phi(x_1, x_2, \ldots, x_n)$ would have the same size as $\phi(1, x_2, \ldots, x_n)$.

   Now suppose that $SAT \in$ **BPP**. Using the boosting argument from class, we can assume that there is a probabilistic polynomial time Turing Machine M that with probability at least $1/2^k$ outputs the correct answer on any input of size k. Run the following algorithm:

   Input: Φ
   Initialize Φ' to be Φ
   For $k = 1, \ldots, n$
   $\Phi' \leftarrow \Phi'$ with x_k set to 0
   Run M with input (Φ', y) using a new random string y
   if M accepts, keep x_k set to 0 in Φ'
   if M rejects, change Φ' by setting x_k to 1 instead of 0
   Check that the current setting for all n variables satisfies the original Φ. If so, accept.
   Reject.

   If Φ is not satisfiable, there will be no setting of the variables that will satisfy Φ, so the algorithm will reject. Suppose that Φ is satisfiable, the probability that a run of M in any particular iteration outputs the incorrect answer ia $1/2^k$, where k is the size of Φ. Note that since the size of Φ' is equal to the size of Φ, the probability of failure does not change in each iteration. Since the size of Φ is at least the number of variables, $1/2^k < 1/2^n$. The probability that any of the n runs of M makes an error is at most $n/2^n < 1/2$. Therefore the algorithm is a randomized algorithm that decides SAT with one-sided error, which means that $SAT \in$ **RP**.

2. Show that if $f(n) \geq n$ and $g(n) \geq n$, are proper functions, then **TIME**$(f(n))$ = **NTIME**$(f(n))$ implies that **TIME**$(f(g(n)))$ = **NTIME**$(f(g(n)))$.

   Consider a language L in **NTIME**$(f(g(n)))$. Define

$$L_{pad} = \{x\#^m \mid x \in L \text{ and } m = g(|x|) - |x|\}.$$

If there is an NTM M that decides L in time $f(g(x))$, then there is an NTM M' that decides L_{pad} in time $f(n)$. Note that for each $x \in L$, the corresponding $x\#^m$ has length exactly $g(|x|)$. On input $x\#^m$, M' can verify that the number of # characters is $g(|x|) - |x|$ and then just simulate M on x.

Since $L_{pad} \in$ **NTIME**$(f(n))$, if **TIME**$(f(n)) =$ **NTIME**$(f(n))$, then $L_{pad} \in$ **TIME**$(f(n))$. This means that there is a (deterministic) TM $\hat{M}$ that decides L_{pad} in time $f(n)$. A deterministic $\bar{M}$ can decide L as follows: on input x, add $g(|x|) - |x|$ # symbols to the end of x. (If the input tape is read-only, this can be done on a separate tape.) Then simulate $\hat{M}$ on $x\#^m$. The time complexity of $\bar{M}$ is $f(g(n))$.

3. Suppose that $L_1, L_2 \in$ **NP**. Is $L_1 \cup L_2$ in **NP**? What about $L_1 \cap L_2$? Prove your answers.

   Let M_1 be an NTM that decides L_1 in polynomial time. Let M_2 be an NTM that decides L_2 in polynomial time.

   Here is a polynomial time NTM M that decides $L_1 \cup L_2$. On input x run M_1. If M_1 is about to accept then M will also halt and accept. If M_1 is about to halt and reject then M will simulate M_2 on input x and will accept or reject according to M_2. If either M_1 or M_2 accepts x, then M has an accepting computation on input x. If neither M_1 not M_2 accepts x, then all of M's computation paths on input x will be rejecting.

   Here is a polynomial time NTM M that decides $L_1 \cap L_2$. On input x run M_1. If M_1 is about to reject then M will also halt and reject. If M_1 is about to halt and accept then M will simulate M_2 on input x and will accept or reject according to M_2. If either M_1 or M_2 rejects x, then all of M's computation paths on input x will be rejecting. If both M_1 and M_2 accept x, then M has an accepting computation on input x.

4. A complexity class $\mathcal{C}$ is closed under complement if $L \in \mathcal{C}$ implies that $co - L \in \mathcal{C}$.

   (a) Are **BPP** or **ZPP** closed under complement? Justify your answer.

   (b) If either **RP** or **co-RP** are closed under complement then which complexity classes (among the ones we have defined in class) are equivalent to **RP**?

   BPP and **ZPP** are both closed under complement. Suppose that $L \in$ **BPP**. Then there is a deterministic Turing Machine M such that if $x \in L$, then $Pr_y[M(x, y)$ accepts$] \geq 2/3$, and if $x \in L$, then $Pr_y[M(x, y)$ accepts$] \leq 1/3$. Consider TM M' that simulates M and just swaps q_{acc} and q_{rej}. Then $x \in L$, then $Pr_y[M'(x, y)$ accepts$] \leq 1/3$, and if $x \in L$, then $Pr_y[M'(x, y)$ accepts$] \geq 2/3$. Therefore M' decides $co - L$ according to the acceptance criteria of **BPP** which means that $co - L \in$ **BPP**.

   Suppose that $L \in$ **ZPP**. Then there is a deterministic Turing Machine M such that for all x, $Pr_y[M(x, y)$ accepts or rejects$] \geq 1/2$, and $Pr_y[M(x, y)$ outputs "Don't Know"$] \leq 1/2$. Furthermore, if M accepts or rejects, then M outpus the correct answer. Consider TM M' that simulates M and just swaps q_{acc} and q_{rej}. Then Therefore M' decides $co - L$ according to the acceptance criteria of **ZPP** which means that $co - L \in$ **ZPP**.

If $\mathbf{RP}$ is closed under complement, then $\mathbf{RP} = \mathbf{co} - \mathbf{RP}$. Since $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co} - \mathbf{RP}$, $\mathbf{RP} = \mathbf{co} - \mathbf{RP}$ implies that $\mathbf{RP} = \mathbf{co} - \mathbf{RP} = \mathbf{ZPP}$.