CS 262: Computational Complexity *Instructor: Sandy Irani*

Homework 4

Due: May 2, 2018

1. Show that the class $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co} - \mathbf{RP}$.

   Suppose that $L \in \mathbf{ZPP}$. There is a **ZPP** Turing Machine M that decides L. Consider a new Turing Machine M' that behaves identially to M, except that if M transitions to the "don't know" state, then M' rejects. If $x \notin L$, then M' will reject x, regardless of whether M succeeds in finding the correct answer for x. If $x \in L$, then M' will output the correct answer if M reaches an answer for x, which happens with probability at least 1/2. Therefore M' is an **RP** decider for L. Similarly, define M" that behaves identially to M, except that if M transitions to the "don't know" state, then M" accepts. By a similar argument M" is a **co-RP** decider for L. Therefore $L \in \mathbf{RP} \cap \mathbf{co} - \mathbf{RP}$.

   Now suppose that $L \in \mathbf{RP} \cap \mathbf{co} - \mathbf{RP}$. Let M be an **RP** decider for L, and let M' be a **co-RP** decider for L. Define a new TM M" that on input x runs M and then M' on input x. If M and M' agree as to whether to accept or reject x, then output that answer. If M and M' disagree as to whether to accept x, then output "don't know". Suppose that $x \in L$. M' will accept x with probability 1 and M will accept x with probability at least 1/2. Therefore, M" will accept x with probability at least 1/2 and will otherwise output "don't know". Similarly for $x \notin L$, M rejects x with probability 1 and M' rejects x with probability at least 1/2. Therefore, M" will reject x with probability at least 1/2 and will otherwise output "don't know". Therefore M" is a **ZPP** decider for L.

2. Describe a *decidable* language that is in **P/poly** but not in **P**.

   Conside a language $L \in \mathbf{TIME}(2^{2^n})$ that is not in $L \in \mathbf{TIME}(2^{n^k})$. The Time Hierarchy Theorem says that there must be such a language. Furthermore, L is decidable since $L \in \mathbf{TIME}(2^{2^n})$. Define U_L to be the unary version of L: $1^n \in U_L$ if and only if the binary representation of n is in L. U_L is in $L \in \mathbf{TIME}(2^{n^k})$ because a TM on input 1^n could translate n into binary (creating an input of size $\log n$) and run the TM that decides L in time $2^{2^{\log n}} = 2^n$.

   U_L is not in P, otherwise there would be a Turing Machine that can decide L in $\mathbf{TIME}(2^{n^k})$. On input x, write n 1's on the worktape, where x is the binary reprsentation of n. Then run the polynomial time algorithm on 1^n. The length of 1^n is $2^{|x|}$ and the running time of the algorithm is polynomial in $2^{|x|}$ which is in $\mathbf{TIME}(2^{n^k})$.

   We will show that U_L is in **P/poly**. The advice for inputs of length n is whether $1^n \in U_L$. On input x of length n, determine if x is all 1's. If so, output whether the advice indicates that $1^n \in U_L$. If not, then reject.

3. A language $L \subseteq \{0,1\}^*$ is *sparse* if there is a polynomial p such that $|L \cap \{0,1\}^n| \leq p(n)$ for all n. Show that every sparse language is in **P/poly**.

   The advice for length n is an enumeration of all the strings x, such that $|x| = n$ and $x \in L$. Since thre are only $p(n)$ such x's, the length of the advice would be $n \cdot p(n)$ which is also polynomial in n. The TM with advice that that decides L on input x would just check whether x is listed in the advice. If so accept, if not reject.

4. The class **P/log** is the class of languages decidable by a Turing Machines running in polynomial time that take O(log n) bits of advice. Show that $SAT \in$ **P/log** implies **P = 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 $SAT \in$ **P/log**. There is a TM M with advice of length $c \cdot \log n$ that solves SAT in polynomial time. Run the following algorithm:

   Input: Φ
   For each y of length $c \cdot \log n$
       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)
           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. There is some y for which M correctly determines whether every Boolean formula that has the same size as Φ is satisfiable. When the for loop reaches that y, every call to M with input (Φ', y) should return the correct answer. (Note that all the Φ' expressions have the same size as Φ, so the same advice y should work for all of them.) The algorithm will find a correct assignment to the variables that satisfies Φ and will accept.