Solutions to Exercise Sheet 3

1. Question: Show that if $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\mathsf{NP} = \mathsf{RP}$. HINT: Use the downward self-reducibility of SAT to eliminate error on NO instances.

   Solution: Assume $\mathsf{NP} \subseteq \mathsf{BPP}$. Note that $\mathsf{RP} \subseteq \mathsf{NP}$ unconditionally, hence we just need to show that $\mathsf{NP} \subseteq \mathsf{RP}$. Since SAT is NP-complete and RP is closed under polynomial-time m-reductions, it is enough to show that SAT is in RP.

   By assumption SAT is in BPP. Let M be a probabilistic Turing machine running in polynomial time and accepting SAT with error at most $1/3$. Using error amplification, we can define a probabilistic Turing machine M' running in polynomial time and accepting SAT with error at most $2^{-\Omega(n)}$, where n is the input length. We will define a probabilistic Turing machine N running in polynomial time, which on input ϕ, accepts with probability at least $1/2$ if ϕ is satisfiable, and accepts with probability 0 if ϕ is unsatisfiable.

   The basic idea is to use downward self-reducibility to construct a satisfying assignment with high probability for YES instances, and to accept only if a satisfying assignment has been constructed. More precisely, N does the following on an input ϕ. Assume wlog that the variables in ϕ are $x_1, x_2 \ldots x_m$. N first runs M on ϕ. If M rejects, N rejects. Otherwise N sets x_1 to "false" in ϕ and runs M on the corresponding formula ϕ_0. If M accepts, N continues to build a satisfying assignment by setting x_2 to "false" in ϕ_0 and running M on the corresponding formula ϕ_{00}. If M rejects, N runs M on formula ϕ_1 obtained by setting x_1 to "true" in ϕ, continuing to build an assignment if M accepts on ϕ_1 and rejecting otherwise. This process continues until either N rejects or all variables are set. If the latter, N checks whether the corresponding assignment satisfies ϕ. If yes, it accepts, otherwise it rejects.

   N runs in polynomial time since it makes at most a linear number of calls to the polynomial-time probabilistic TM M, and each of these calls is on an input whose length is at most the length of ϕ (setting variables can only decrease the length of a formula). N never accepts on an unsatisfiable formula, hence all we need to show is that N accepts with probability at least $1/2$ on a satisfiable formula. Note that if M always gives correct answers on calls to M, then when ϕ is satisfiable, N constructs a satisfying assignment to ϕ and hence accepts. The probability that this happens is at least $1 - n2^{-\Omega(n)}$, which is at most $1/2$ for large enough n, since the probability that M gives at least one wrong answer is at most $n2^{-\Omega(n)}$ by the union bound.

2. Question: Indecisive Turing machines are Turing machines which, in addi-

tion to accepting and rejecting states, have a “don’t know” state in which the computation may terminate. A language L is said to be in ZPP (zero-error probabilistic polynomial time) if there is an indecisive randomized Turing machine M halting in polynomial time such that:

(a) If $x \in L$, M does not halt in a rejecting state on *any* computation path (it halts either in an accepting state or the “don’t know” state), and it halts in an accepting state with probability at least $1/2$.

(b) If $x \notin L$, M does not halt in an accepting state on *any* computation path (it halts either in a rejecting state or the “don’t know” state), and it halts in a rejecting state with probability at least $1/2$.

Prove that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

Solution: Note that when asked to show an equality between two complexity classes, you need to show two things: that the first complexity class is contained in the second, and that the second complexity class is contained in the first.

We first show that $\mathsf{ZPP} \subseteq \mathsf{RP} \cap \mathsf{coRP}$. This is the easier part of the argument. We simply show that $\mathsf{ZPP} \subseteq \mathsf{RP}$, and from the fact that ZPP is closed under complement (which can be seen just from switching acceptance and rejection in the definition), we also get that $\mathsf{ZPP} \subseteq \mathsf{coRP}$.

Let $L \in \mathsf{ZPP}$. Then there is an indecisive Turing machine M' running in polynomial time and deciding L, as per the definition. We define a machine M' which is a randomized Turing machine in the usual sense and witnesses that $L \in \mathsf{RP}$. M' is the same as M, except that now “don’t know” states are also labelled as accepting. Now we have that if $x \in L$, M accepts with probability $1/2$ and if $x \notin L$, M accepts with probability 0. Moreover, M runs in polynomial time (since M' does). Thus $L(M) = L \in \mathsf{RP}$.

Next we show that $\mathsf{RP} \cap \mathsf{coRP} \subseteq \mathsf{ZPP}$. Let $L \in \mathsf{RP} \cap \mathsf{coRP}$. Let M be a randomized polynomial-time Turing machine witnessing that $L \in \mathsf{RP}$ and M' be a randomized polynomial-time Turing machine witnessing that $L \in \mathsf{coRP}$. We define an indecisive polynomial-time machine N witnessing that $L \in \mathsf{ZPP}$.

Given input x, N simulates both M and M' on x. If M accepts, then N accepts. If M' rejects, then N rejects. If M rejects and M' accepts, then N halts in a “don’t know” state.

Clearly N is polynomial-time. If $x \in L$, then M accepts with probability at least $1/2$, and so by definition of N, N accepts with probability at least $1/2$ as well. Moreover, if $x \in L$, M' accepts with probability 1, hence N rejects with probability 0, i.e., it always halts in either an accepting state or a “don’t know” state.

If $x \notin L$, then M' rejects with probability at least $1/2$, so N rejects with probability at least $1/2$. Moreover, if $x \notin L$, M accepts with probability 0, so N never accepts, i.e., N always halts either in a rejecting state or a "don't know" state.

Thus N witnesses that $L \in \mathsf{ZPP}$.

3. Question: $\mathsf{PCP}[r(n), q(n)]$ is the class of languages accepted by probabilistically checkable proof systems where the verifier uses at most $r(|x|)$ random bits and makes at most $q(|x|)$ non-adaptive queries to the proof on any input x. Show that $\mathsf{PCP}[0, \log(n)] = \mathsf{P}$.

   Solution: We first show that $\mathsf{P} \subseteq \mathsf{PCP}[0, \log(n)]$, and then the converse.

   Let $L \in \mathsf{P}$ and let M be a deterministic polynomial-time machine decides L. We define a probabilistically checkable proof system with no randomness and 0 queries deciding L. We define the verifier V for this proof system as follows: V does not access the proof at all, instead it simulates M on x, accepting if M accepts and rejecting otherwise. If $x \in L$, then there exists a proof such that V accepts with probability 1 (indeed this is true irrespective of the proof), and if $x \notin L$, then for all proofs V rejects with probability 1, showing that $L \in \mathsf{PCP}[0, 0] \subseteq \mathsf{PCP}[0, \log(n)]$.

   The harder part is showing that $\mathsf{PCP}[0, \log(n)] \subseteq \mathsf{P}$. Let $L \in \mathsf{PCP}[0, \log(n)]$. This means that there is a proof system with a polynomial-time verifier V using no randomness and making at most $\log(n)$ non-adaptive queries on any input x of length n which decides L. We use V to define a polynomial-time Turing machine M deciding L.

   Note that V uses no randomness, therefore it either simply accepts or simply rejects. Moreover, whether it accepts or rejects is purely a function of the input x and the at most $\log(n)$ proof bits that V reads. If there were a proof for which V accepted, then there would be some $(0, 1)$-assignment to these proof bits for which V would accept. And if V were to reject for every proof, then no $(0, 1)$-assignment to the proof bits could cause V to reject.

   We define M as follows. Given input x of length n, M simply searches over all possible assignments to the at most $\log(n)$ proof bits accessed by V on input x, and checks whether V accepts for any of these assignments. If yes, it accepts, otherwise it rejects. The time complexity of M arises from the exhaustive search over assignments, and the complexity of simulating V. The first is polynomial-time since there are at most $\log(n)$ proof bits to be considered and hence at most n assignments; the second is polynomial-time since V is polynomial-time. M accepts exactly those inputs x accepted by the proof system, hence M decides L correctly.

Rahul Santhanam, Apr 2013