Solution

Computational Complexity – Homework 7

Discussed on 21.05.2019.

Exercise 7.1

Show that, if $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{RP} = \mathbf{NP}$.

Solution: It is known to be the case that $\mathbf{RP} \subseteq \mathbf{NP}$. (Simply forget the probability distribution over the certificates and one ends up with the definitoin of **NP**!)

Note that in case of **NP** we are only promised (at least) one certificate u such that $M_1(x, u = 1)$. That is the probability of picking a witnessing certificate uniformly at random is just $2^{-p(|x|}$. Thus if $x \in L$, we have $P[A_{x,M}] \geq 2^{-p(|x|)}.(3/4)$.

So now suppose that $\mathbf{NP} \subseteq \mathbf{BPP}$. We need to show that $\mathbf{NP} \subseteq \mathbf{RP}$.

As $\mathbf{NP} \subseteq \mathbf{BPP}$, we can consider a polynomial-time Turing machine M that solves SAT correctly with probability more than $1 - \frac{1}{(n+1)^2}$ where n is the number of variables. But we can find the minimal satisfying assignment of a satisfiable formula but asking $n + 1$ questions about satisfiability We ask whether the the formula itself is satisfiable; if it is, we compute lexicographically minimal prefixes of satisfying assignments of growing length. If $x_1 = 0$ can be extended to satisfyign assignment, it is the minimal prefix of length 1, otherwise all satisfying assignments start with $x_1 = 1$. Then we check if fixing $x_2 = 0$ (using the chosen value of x_1) can be extended to satisfying assignment; if yes, the minimal assignment is obtained by fixing $x_2 = 0$ and extending further, otherwise it must have $x_2 = 1$. And so on. In the end we can easily check if the output is indeed a satisfying assignment.

Using M to generate a satisfying assignment then verifying the output accepts each formula from SAT with probability at least $1 - 1/n$ (the probability of any mistake is at most $(n + 1)\frac{1}{(n+1)^2}$ as we make $n + 1$ queries and can apply the union bound) and definitely rejects each formula not in SAT. This matches the definition of **RP**, so $SAT \in \mathbf{RP}$ and as SAT is **NP**-complete we obtain $\mathbf{NP} \subseteq \mathbf{RP}$.

Exercise 7.2

Show that

(a) **RP**, **BPP**, and **PP** are closed under $\leq_p$.

Remark: Recall that a class **C** is closed under $\leq_p$ if $A \leq_p B \wedge B \in \mathbf{C} \Rightarrow A \in \mathbf{C}$.

(b) **RP** and **BPP** are closed under intersection and union.

Exercise 7.3

A *probabilistic alternating* Turing machine (short: PATM) is a tuple $(Q_{\frac{1}{2}}, Q_{\exists}, \Gamma, \delta_0, \delta_1)$ where

- $Q := Q_{\frac{1}{2}} \cup Q_{\exists}$ is the set of control states. ($Q_{\frac{1}{2}}$ and $Q_{\exists}$ are required to be disjoint.)
- Γ is the alphabet.
- δ_0, δ_1 are two transition functions.

A run of a PATM $M = (Q_{\frac{1}{2}}, Q_{\exists}, \Gamma, \delta_0, \delta_1)$ on a given input x is simply a run by the underlying NDTM defined by $(Q_{\frac{1}{2}} \cup Q_{\exists}, \Gamma, \delta_0, \delta_1)$. In particular, M runs in time $T(n)$ if every run on input x takes time at most $T(|x|)$, i.e., the computation tree of M on input x has height at most $T(|x|)$. (Recall the inductive definition of configuration tree: starting from the initial configuration on input x (the root), every inner node of the tree is a non-halting configuration c of M which has exactly two childrens $\delta_0(c)$ and $\delta_1(c)$, even if $\delta_0(c) = \delta_1(c)$.)

The intuition of a PATM is that it combines randomization with nondeterminism: in a configuration with a control state contained in $Q_{\exists}$ a PATM basically explores both possible successors in parallel, while in a configuration with control state in $Q_{\frac{1}{2}}$ it chooses on of the two possible successors uniformly at random. More formally, the probability that M accepts x ($\Pr[M(x) = 1]$) is then defined by labeling the computation tree bottom-up as follows:

- A leaf is labeled by 1 if it corresponds to a accepting configuration, otherwise it is labeled by 0.
- An inner node which corresponds to a control state from $Q_{\frac{1}{2}}$ is labeled by the average of the labels of its two children;
- while an inner node corresponding to a control state from $Q_\exists$ is labeled by the maximum of its two children.

The label of the root of the computation tree of M on input x is then the probability that M accepts x, short $\Pr[M(x) = 1]$. Similarly, $\Pr[M(x) = 0] := 1 - \Pr[M(x) = 1]$.

(a) Show that for every poly-time PATM M there is a poly-time PATM N s.t.:

- $\Pr[M(x) = 1] = \Pr[N(x) = 1]$ for all $x \in \{0, 1\}^*$.
- Every run of N on a given input x takes time exactly $2|x|^k$ for some $k > 0$.
- Every inner node with control state in $Q_{\frac{1}{2}}$ ($Q_\exists$) has only children with control state in $Q_\exists$ ($Q_{\frac{1}{2}}$).

(b) Let $M = (Q_\exists, Q_\forall, \Gamma, \delta_0, \delta_1)$ be a poly-time ATM deciding the language L. We can reinterpret M also a PATM by setting $Q_{\frac{1}{2}} := Q_\forall$. Show that

$$x \in L \Leftrightarrow \Pr[M(x) = 1] = 1.$$

(c) The class **APP** is defined as follows:

A language L is contained in **APP** if there is a poly-time PATM M s.t.

$$x \in L \Leftrightarrow \Pr[M(x) = 1] \geq 3/4.$$

- Show that **APP** $\subseteq$ **PSPACE** by adapting the **PSPACE**-algorithm for deciding QSat.
- Show that **PSPACE** $\subseteq$ **APP** by adapting the proof of **NP** $\subseteq$ **PP** given in the lecture.

  Hint: Recall that **AP** = **PSPACE**, i.e., for every $L \in$ **PSPACE** there is a poly-time alternating Turing machine deciding L. Now copy the construction from the proof of **NP** $\subseteq$ **PP** in order to obtain from a poly-time ATM a poly-time PATM M with $x \in L \Leftrightarrow \Pr[M(x) = 1] \geq 3/4$.

(d) The class **ABPP** is defined as follows:

A language L is contained in **ABPP** if there is a poly-time PATM M s.t.

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[M(x) = 1] \leq 1/4.$$

Obviously, we have **ABPP** $\subseteq$ **APP**.

- Show that **ABPP** = **IP** = **APP** = **PSPACE**.

  Hint: You already know **ABPP** from the lecture by some other name.

(e) Assume we extend the definition of PATMs by partitioning the control states into three classes $Q_{\frac{1}{2}}, Q_\exists, Q_\forall$; the acceptance probability $\Pr[M(x) = 1]$ is defined as above where the value of a node corresponding to a control state of $Q_\forall$ is defined to be the minimum of the values of its two children. Call such a Turing machine a PAATM.

- Using PAATMs define the complexity classes **AAPP** and **AABPP** analogously to **APP** and **ABPP**.

  Discuss how these relate to **APP**, **PP**, **ABPP**, **BPP**, **AP**, **PSPACE**, **IP**, **AM**.