03684155: On the P vs. BPP problem. 4/12/2016 – Lecture 6

The easy witness method

Amnon Ta-Shma and Dean Doron

Suppose there exists a language $L \in \mathsf{NEXP} \setminus \mathsf{EXP}$. Let $M(x,y)$ be a non-deterministic TM solving L in NEXP. We now ask the following question: Suppose someone gives us an input $x \in L$. We know that there exists a witness y such that $M(x,y)=1$. How difficult it is to find such a witness y?

One thing that we can say for sure is that there exists an x (in fact, an infinite sequence of inputs) for which there is no easy witness, where a sequence of witnesses is easy if it can be described by a uniform family of polynomial size circuits. This is true, because otherwise there is always an easy witness, and therefore the procedure that checks all the easy witness will solve L in EXP, in contradiction to the fact that $L \notin \mathsf{EXP}$.

However, having that, we can use the fact that there are no easy witnesses as a *hardness* proof! Namely, if we are given an x for which there exists a witness y with $M(x,y)=1$, while there are no *easy* witnesses y, then every witness y is necessarily a truth table of a function hard against polynomial size circuits. Therefore, as we saw in previous lectures, we can use it for our own good and use its hardness to construct a PRG against polynomial-sized circuits.

Said differently, we encounter a win-win scenario. Either every language in NEXP solvable by $M(x,y)$ also has (possibly except for finitely many inputs) easy witnesses and then $\mathsf{NEXP} = \mathsf{EXP}$, or else there is an infinite sequence of inputs x_i, such that x_i has some witness and every witness y for x_i is necessarily a hard function. In such a case we have PRGs. The approach is called the "easy witness method".

Of course, things are not as easy as that. First, we have the annoying (but usually harmless) infinitely-often directive. Also, we need someone to give us the (infinitely many) inputs x_i-s that have a witness but no easy witnesses. Thus, non-uniformity enters the picture.

1 $\mathsf{NEXP} \neq \mathsf{EXP}$ implies $\mathsf{MA} \subseteq \text{io-}\mathsf{NTIME}(2^{n^a})/n$

Theorem 1 ([1])**.** *If* $\mathsf{NEXP} \neq \mathsf{EXP}$ *then there exists a fixed constant a such that* $\mathsf{MA} \subseteq \text{io-}\mathsf{NTIME}(2^{n^a})/n$.

Proof. Fix a language $A_0 \in \mathsf{NEXP} \setminus \mathsf{EXP}$. Then A_0 is decidable by a TM $A_0(x,y)$ in time $T = 2^{n^{a_0}}$, so we can also assume that $y \in \{0,1\}^T$.

Intuitively, we would like to build another Turing Machine AE that operates like A_0, but instead of guessing the witness y, tries all easy witnesses y that are described by a small circuit. We identify a circuit C on ℓ inputs with an assignment $\{0,1\}^{2^\ell}$, by letting the value of the $(i_1,\ldots,i_\ell)$ bit be $C(i_1,\ldots,i_\ell)$. We say a witness $y \in \{0,1\}^T$ is easy, if it is represented by a small circuit of size polynomial in $\log(T)$.

We take $AE_{s(n)}$ be the TM that checks all possible *easy* witnesses. Specifically, on input $x \in \{0,1\}^n$, $AE_{s(n)}$ goes over all Boolean circuits with n^{a_0} inputs and size at most $s(n)$, and for each such circuit C the machine simulates $N(x,(C(0^{n^{a_0}}),\ldots,C(1^{n^{a_0}})))$. $AE_{s(n)}$ accepts iff the simulation accepts for some C. Note that $AE_{s(n)}$ runs in deterministic time $s^{O(s)} \cdot 2^{O(n^{a_0})}$. As $A_0 \notin \mathsf{EXP}$ we may conclude that for *every* constant c, AE_{n^c} does not solve A_0.

If $x \notin A_0$, $AE_{n^c}(x)$ necessarily rejects x as it should. Hence for every c there exits an infinite sequence $N_c \subseteq \mathrm{N}$ and corresponding inputs $X_c = \{x_n \in \{0,1\}^n \mid n \in N_c\}$, such that for every $n \in N_a$, $x_n \in A_0$ but $AE_{n^c}(x) = 0$.

With that we prove:

Lemma 2. $\mathsf{MA} \subseteq \text{io-}\mathsf{NTIME}(2^{n^a})/n$.

Proof. Let $B \in \mathsf{MA}$. Then, there exists a TM $M(x, \gamma, z)$ and a constant b such that

- If $x \in M$, there exists γ such that $\Pr_y[M(x, \gamma, y) = 1] \geq \frac{2}{3}$, and,
- If $x \in M$, there for all γ, $\Pr_y[M(x, \gamma, y) = 1] \leq \frac{2}{3}$.

Furthermore, $\gamma \in \{0,1\}^{n^b}$, $z \in \{0,1\}^{n^b}$ are the random coins and $M(x, \gamma, y)$ is computed in n^b time. We want to derandomize M for infinitely-many lengths of x.

Fix $c = 10b$. Let M' be a nondeterministic TM with advice, that on input length n gets the advice $x_n \in X_c$ (if $n \notin N_a$ the advice is arbitrary). M' does the following:

- It first guesses $y \in \{0,1\}^{2^{n^{a_0}}}$ such that $A_0(x_n, y) = 1$. We view $y \in \{0,1\}^{2^{n^{a_0}}}$ as the truth table of a function $f_y : [2^{n^{a_0}}] \to \{0,1\}$. We identify $[2^{n^{a_0}}]$ with $\{0,1\}^{n^{a_0}}$ and in this notation $f_y : \{0,1\}^{\ell = n^{a_0}} \to \{0,1\}$. Notice that we know that $Size(f_y) \geq n^c$.
- It takes the $PRG = PRG^f : \{0,1\}^{\ell^2 = n^{2a_0}} \to \{0,1\}^{n^b}$ that fools circuits of size n^b, runs in time $2^{O(\ell)}$ and works as long as $Size(f) \geq n^c$ (here we take the NM generator with constant intersection size designs).

B' then guesses a witness $\gamma \in \{0,1\}^{n^b}$ and simulates $B(x, \gamma, z)$, over all z in the image of $PRG^{f_y}(U_{\ell^2})$. It decides according to the majority vote.

It then follows that B' solves B correctly for every input of length that is in N_c. Also, B' runs in $\mathsf{NTIME}(2^{O(\ell^2)}) = \mathsf{NTIME}(2^{O(n^{2a_0})})$ and uses n bits of advice. Thus, $B \in \text{io-}\mathsf{NTIME}(2^{n^a})/n$. □

□

2 NEXP ⊆ P/poly implies NEXP = MA

Theorem 3. $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ *implies* $\mathsf{NEXP} = \mathsf{MA}$.

Proof. Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$ we have $\mathsf{EXP} = \mathsf{MA}$. We claim that we must have $\mathsf{NEXP} = \mathsf{EXP}$. Suppose not. Then, there exists a fixed constant a such that $\mathsf{NEXP} \neq \mathsf{EXP}$ hence $\mathsf{EXP} = \mathsf{MA} \subseteq \text{io-}\mathsf{NTIME}(2^{n^a})/n$. However this contradicts the theorem we have obtained before (using diagonalization). Hence, $\mathsf{NEXP} = \mathsf{EXP} = \mathsf{MA}$. □

References

[1] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.