03684155: On the P vs. BPP problem. 30/12/2016 – Lecture 11

Promise problems

Amnon Ta-Shma and Dean Doron

1 Definitions and examples

In a *promise problem*, we are interested in solving a problem only on a subset of the inputs, which is a very natural phenomenon ("given a *connected* graph, determine whether or not..."). Formally:

Definition 1. *A* promise problem Π *is a pair* (Π_Y, Π_N) *such that* $\Pi_Y, \Pi_N \subseteq \{0,1\}^\star$ *and* $\Pi_Y \cap \Pi_N = \emptyset$*. The set* $\Pi_Y \cup \Pi_N$ *is called the promise.*

Given a complexity class $\mathcal{C}$, we denote the class Promise-$\mathcal{C}$ to denote the set of promise problems decidable via the same resources as problems in $\mathcal{C}$. For example, $\Pi = (\Pi_Y, \Pi_N) \in$ Promise-BPP if there exists a probabilistic polynomial-time algorithm M such that:

- For every $x \in \Pi_Y$ it holds that $\Pr_y[M(x,y) = 1] \geq 2/3$.
- For every $x \in \Pi_N$ it holds that $\Pr_y[M(x,y) = 0] \geq 2/3$.

Note that we do not impose any requirements on the output of M on x-s outside the promise (but do, however, require M to run in polynomial-time).

Reductions also extend naturally to promise problems.

Definition 2. *The promise problem* $\Pi = (\Pi_Y, \Pi_N)$ *is (Karp) reducible to the probmise problem* $\Pi' = (\Pi'_Y, \Pi'_N)$ *if there exists a polynomial-time computable function* f *such that:*

- *For every* $x \in \Pi_Y$ *it holds that* $f(x) \in \Pi'_Y$*.*
- *For every* $x \in \Pi_N$ *it holds that* $f(x) \in \Pi'_N$*.*

Definition 3. *The promise problem* $\Pi = (\Pi_Y, \Pi_N)$ *is Cook-reducible to the probmise problem* $\Pi' = (\Pi'_Y, \Pi'_N)$ *if there exists a polynomial-time oracle TM* M *such that:*

- *For every* $x \in \Pi_Y$ *it holds that* $M^{\Pi'}(x) = 1$*.*
- *For every* $x \in \Pi_N$ *it holds that* $M^{\Pi'}(x) = 0$*.*

Although it is unknown whether or not BPP has a complete problem, it follows immediately that the above problem is complete for Promise-BPP (under Karp reductions): The Yes instances are Boolean circuits that evaluate to 1 on at least $2/3$ fraction of their inputs whereas the No instances are Boolean circuits that evaluate to 0 on at least $2/3$ fraction of their inputs.

A more notorious example that differentiates promise problems from their decision problems counterparts is the existence of a Promise-NP-hard problem (under Cook reductions) in Promise-NP $\cap$ Promise-coNP. An analogous result in the non-promise world would imply NP = coNP. The promise problem PSAT is the following:

- Yes instances: (φ_1, φ_2) such that $\varphi_1 \in \text{SAT}$ and $\varphi_2 \notin \text{SAT}$.
- No instance: (φ_1, φ_2) such that $\varphi_1 \notin \text{SAT}$ and $\varphi_2 \in \text{SAT}$.

The following fact is easy:

Claim 4. $\text{PSAT} \in \text{Promise-NP} \cap \text{Promise-coNP}$.

Furthermore:

Claim 5. Promise-NP *is Cook-reducible to* PSAT.

Proof. The reduction from SAT to PSAT is the following: On input φ with n variables and oracle access to PSAT, the TM M proceeds as follows:

1. For every $i \in [n]$,
   (a) Let $\varphi_{i,b}$ be the formula obtained from φ by setting x_i to b.
   (b) Invoke Π' on $(\varphi_{i,1}, \varphi_{i,0})$. If the answer is Yes then set $x_i = 1$ in φ and otherwise set $x_i = 0$ in φ.
2. Accept iff $\varphi = 1$.

If φ is not satisfiable then of course we will always reject. Note that at any stage, if φ is satisfiable and the query $(\varphi_{i,0}, \varphi_{i,1})$ is answered with b then $\varphi(x_i = b)$ is satisfiable. Thus, if φ is satisfiable we will end up with 1 and accept. □

How did we "gain" this power? The promiscuous way in which we applied the oracle calls does not maintain the standard meaning of the reduction – we made queries that violate the promise, but still we have shown that the reduction remains valid regardless of the answers given to these violating queries. However, these queries fail to preserve the structural consequences we would expect.

One may eliminate the problems arising from such queries by requiring that the reduction does not make them (we will see this in the exercise). Also, Karp reductions are such an example.

2 Another look at $\text{BPP} \subseteq \Sigma_2$

2.1 $\text{BPP} \subseteq \Sigma_2$

We show a different way of proving Sipser's theorem $\text{BPP} \subseteq \Sigma_2$, via a randomness-efficient error-reduction, due to Goldreich and Zuckerman [1]. Recall that since $\text{BPP} \subseteq \Sigma_2$, $\text{P} = \text{NP}$ implies $\text{P} = \text{BPP}$.

The obvious way to amplify the success probability is by repeated independent sampling. This way, we can transform a constant-error BPP machine that uses m coins to a one with 2^{-t} error using $O(mt)$ coins. Using *extractors*, we know how to do it using $O(t)$ coins. What we should note is that the number of coins used is essentially the logarithm of the error bound. Specifically:

Theorem 6 ([2])**.** *For every $L \in \mathsf{BPP}$ there exists a probabilistic TM M and a polynomial p such that*

$$\Pr_{y \in \{0,1\}^{p(|x|)}}[M(x,y) \neq L(x)] \;<\; 2^{-2p(|x|)/3}.$$

The interested reader may refer to last year's course for the (non-tight) proof: `http://www.cs.tau.ac.il/~amnon/Classes/2016-PRG/Lecture2.pdf`.

Theorem 7. $\mathsf{BPP} \subseteq \Sigma_2$.

Proof. Let $L \in \mathsf{BPP}$, so there exists a polynomial-time TM M and a polynomial p such that if $x \in L$ then $\Pr_y[M(x,y) = 1] > 1 - 2^{-2p(|x|)/3}$ and if $x \notin L$ then $\Pr_y[M(x,y) = 1] < 2^{-2p(|x|)/3}$, where $y \in \{0,1\}^{p(|x|)}$. On input x, let $m = p(|x|)$.

We know that if $x \in L$ then there are at most $2^{m/3}$ possible y-s for which $M(x,y) = 0$. Thus, there are at most $2^{m/3}$ prefixes $y' \in \{0,1\}^{m/2}$ for which some y'' exists so that $M(x,y'y'') = 0$. If $x \notin L$ then there are at most $2^{m/3}$ possible y-s for which $M(x,y) = 1$, so for each $y' \in \{0,1\}^{m/2}$, it holds that

$$\Pr_{y'' \in \{0,1\}^{m/2}}[M(x,y'y'') = 1] \;\leq\; \frac{2^{m/3}}{2^{m/2}} \;=\; o(1).$$

Consequently, if $x \in L$ then

$$\exists y' \; \forall y'' \; M(x,y'y'') = 1,$$

and if $x \notin L$ then

$$\forall y' \; \exists y'' \; M(x,y'y'') = 0.$$

Thus, $L \in \Sigma_2$. In fact, it shows an even stronger claim: we can replace the $\exists$ quantifier with a *for almost all* quantifier. □

What we did is essentially divided the set of 2^m possible coin tosses into $2^{m/2}$ subsets of size $2^{m/2}$. To handle the $x \in L$ case, we used the fact that the number of bad coin tosses is smaller than the number of subsets. To handle the $x \notin L$ case, we used the fact that the number of bad coin tosses is smaller than the size of each subset. Both facts are non-trivial, and are a consequence of the efficient error-reduction scheme. The above prove also holds for Promise-BPP.

2.2 P, Promise-RP and Promise-BPP

As far as we know, it may be the case that $\mathsf{RP} = \mathsf{P}$, yet $\mathsf{BPP} \neq \mathsf{P}$. In the promise world, this is not the case.

Theorem 8. *If* $\mathsf{Promise\text{-}RP} = \mathsf{P}$ *then* $\mathsf{Promise\text{-}BPP} = \mathsf{P}$.

Here, the more standard $\mathsf{BPP} \subseteq \Sigma_2$ proof will be useful. A closer look at the proof reveals that is also shows:

Theorem 9. $\mathsf{Promise\text{-}BPP} \subseteq \mathsf{Promise\text{-}RP}^{\mathsf{Promise\text{-}RP}}$.

From this, Theorem 8 easily follows: If $\mathsf{Promise\text{-}RP} = \mathsf{P}$ then $\mathsf{Promise\text{-}BPP} \subseteq \mathsf{P}^{\mathsf{P}} = \mathsf{P}$.

Proof. (of Theorem 8). We prove the stronger claim, that $\mathsf{BPP} \subseteq \mathsf{RP}^{\mathsf{Promise\text{-}RP}}$, even with one oracle call. Let $L \in \mathsf{BPP}$, equipped with a TM M that on input of length n errs with probability at most 2^{-n} and uses at most $p(n)$ random coins for some fixed polynomial p.

Define:

$$B = \left\{(x, y_1, \ldots, y_{p(|x|)}) \mid \exists w,\ M(x, w \oplus y_1) = 0 \wedge \ldots \wedge M(x, w \oplus y_{p(|x|)}) = 0\right\}.$$

Consider an RP^B algorithm, that on input $x \in B^n$, chooses $y_1, \ldots, y_{p(n)}$ independently at random and accepts iff $(x, y_1, \ldots, y_{p(n)}) \notin B$.

If $x \in L$ then for a fixed w and for every i, $\Pr[M(x, w \oplus y_i) = 0] \leq 2^{-n}$. As the choices are independent, $\Pr[\forall i\ M(x, w \oplus y_i) = 0] \leq 2^{-n \cdot p(n)}$. By the union bound over all w-s, we have that $\Pr[x \in B] \leq 2^{-n}$.

Suppose now that $x \notin L$. Fix some $y_1, \ldots, y_{p(n)}$ and i. If we choose w at random, then again, $\Pr[M(x, w \oplus y_i) = 1] \leq 2^{-n}$. By the union bound, $\Pr[\exists i\ M(x, w \oplus y_i) = 1] \leq p(n) \cdot 2^{-n} \leq 1/2$. Thus, for *every* $y_1, \ldots, y_{p(n)}$, there exists a good w and $(x, y_1, \ldots, y_{p(n)}) \in B$. Thus, indeed, $L \in \mathsf{RP}^B$.

The next step is to show that $L \in \mathsf{RP}^{\mathsf{Promise\text{-}RP}}$. Let $C \subseteq B$ and we only know that for a tuple $(x, y_1, \ldots, y_{p(n)})$:

- If more than half of the w-s satisfy $M(x, w \oplus y_1) = 0 \wedge \ldots \wedge M(x, w \oplus y_{p(|x|)}) = 0$ then $(x, y_1, \ldots, y_{p(n)}) \in C$.
- If there is not w-s satisfying $M(x, w \oplus y_1) = 0 \wedge \ldots \wedge M(x, w \oplus y_{p(|x|)}) = 0$ then $(x, y_1, \ldots, y_{p(n)}) \notin C$.

The same proof that $L \in \mathsf{RP}^B$ also shows that $L \in \mathsf{RP}^C$ (verify this). What did we gain? $B \in \mathsf{NP}$, however clearly, $C \in \mathsf{Promise\text{-}RP}$ and we are done. □

3 $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$ revisited

We know that if $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$ then we have a partial derandomization of BPP, namely $\mathsf{BPP} \subseteq \mathsf{io\text{-}SUBEXP}$. The same derandomization also holds for $\Pr\mathsf{BPP}$. This result readily holds for $\mathsf{Promise\text{-}BPP}$, that is $\mathsf{Promise\text{-}BPP} \subseteq \mathsf{io\text{-}SUBEXP}$. However, it is not known whether the existence of hard Boolean functions in EXP is actually necessary for partial derandomization of BPP.

On the other hand, any partial (even nondeterministic) derandomization of $\mathsf{Promise\text{-}BPP}$ yields a circuit lower bound for NEXP:

Theorem 10. *If* $\mathsf{Promise\text{-}BPP} \subseteq \mathsf{Promise\text{-}SUBEXP}$ *then* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

We will give this as an exercise.

Thus, unconditional results in derandomization require either making a distinction between BPP and $\mathsf{Promise\text{-}BPP}$, or proving circuit lower bounds for NEXP. Indeed, we know that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ implies $\mathsf{MA} = \mathsf{NEXP}$, and hence no derandomization of MA is possible unless there are hard functions in NEXP. Since derandomizing $\mathsf{Promise\text{-}BPP}$ also allows to derandomize MA, we conclude that no full derandomization is possible without assuming (or proving) lower bounds for NEXP.

References

[1] Oded Goldreich and David Zuckerman. Another proof that BPP⊆PH (and more). In *Electronic Colloquium on Computational Complexity*, 1997.

[2] David Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4-5):367–391, 1996.