Notes on Computer Theory Last updated: November, 2015

$\mathcal{BPP}$, Proof systems.

Notes by Jonathan Katz, lightly edited by Dov Gordon.

1 Randomized Time Complexity

1.1 How Large is $\mathcal{BPP}$?

We know that

$$\mathcal{P} \subseteq \mathcal{ZPP} = \mathcal{RP} \cap \mathsf{co}\mathcal{RP} \subseteq \mathcal{RP} \cup \mathsf{co}\mathcal{RP} \subseteq \mathcal{BPP} \subseteq \mathsf{PSPACE} \subset .$$

We currently do not have a very good unconditional bound on the power of $\mathcal{BPP}$ — in particular, it could be that $\mathcal{BPP} = \mathsf{NEXP}$. Perhaps surprisingly, especially in light of the many randomized algorithms known, the current conjecture is that $\mathcal{BPP}$ is *not* more powerful than $\mathcal{P}$. We will return to this point later in the semester when we talk about derandomization.

What (unconditional) upper bounds *can* we place on $\mathcal{BPP}$? Interestingly, we know that it is not more powerful than polynomial-size circuits; actually, the following theorem is also a good illustration of the power of non-uniformity.

Theorem 1 $\mathcal{BPP} \subset \mathcal{P}/\mathsf{poly}$.

Proof Let $L \in \mathcal{BPP}$. Using amplification, we know that there exists a polynomial-time Turing machine M such that $\Pr[M(x) \neq \chi_L(x)] < 2^{-|x|^2}$. Say M uses (at most) $p(|x|)$ random coins for some polynomial p. (Note that p is upper-bounded by the running time of M.) An equivalent way of stating this is that for each n, and each $x \in \{0,1\}^n$, the set of "bad" coins for x (i.e., coins for which $M(x)$ outputs the wrong answer) has size at most $2^{p(n)} \cdot 2^{-n^2}$. Taking the union of these "bad" sets over all $x \in \{0,1\}^n$, we find that the total number of random coins which are "bad" for *some* x is at most $2^{p(n)} \cdot 2^{-n} < 2^{p(n)}$. In particular, there exists at least one set of random coins $r_n^* \in \{0,1\}^{p(n)}$ that is "good" for *every* $x \in \{0,1\}^n$ (in fact, there are many such random coins). If we let the sequence of "advice strings" be exactly $\{r_n^*\}$ (using the alternate definition of $\mathcal{P}_{/\mathsf{poly}}$), we obtain the result of the theorem. ■

We can also place $\mathcal{BPP}$ in the polynomial hierarchy:

Theorem 2 $\mathcal{BPP} \subseteq \Sigma_2 \cap \Pi_2$.

Proof We show that $\mathcal{BPP} \subseteq \Sigma_2$; since $\mathcal{BPP}$ is closed under complement, this proves the theorem.

We begin by proving some probabilistic lemmas. Say $S \subseteq \{0,1\}^m$ is *large* if $|S| \geq (1 - \frac{1}{m})2^m$, and is *small* if $|S| < \frac{2^m}{m}$. For a string $z \in \{0,1\}^m$ define $S \oplus z \stackrel{\text{def}}{=} \{s \oplus z \mid s \in S\}$.

Claim 3 *If S is small, then for all $z_1, \ldots, z_m \in \{0,1\}^m$ we have $\bigcup_i (S \oplus z_i) \neq \{0,1\}^m$.*

This follows easily since

$$\left|\bigcup_i (S \oplus z_i)\right| \;\leq\; \sum_i |S \oplus z_i| \;=\; m \cdot |S| \;<\; 2^m.$$

Claim 4 *If S is large, then there exist $z_1, \ldots, z_m \in \{0,1\}^m$ such that $\bigcup_i (S \oplus z_i) = \{0,1\}^m$.*

In fact, we show that choosing at random works with high probability; i.e.,

$$\Pr_{z_1,\ldots,z_m \in \{0,1\}^m}\left[\bigcup_i (S \oplus z_i) = \{0,1\}^m\right] \geq 1 - \left(\frac{2}{m}\right)^m.$$

To see this, consider the probability that some fixed y is not in $\bigcup_i (S \oplus z_i)$. This is given by:

$$\begin{aligned}\Pr_{z_1,\ldots,z_m \in \{0,1\}^m}\left[y \notin \bigcup_i (S \oplus z_i)\right] &= \prod_i \Pr_{z \in \{0,1\}^m}[y \notin (S \oplus z)] \\ &\leq \left(\frac{1}{m}\right)^m.\end{aligned}$$

Applying a union bound by summing over all $y \in \{0,1\}^m$, we see that the probability that there *exists* a $y \in \{0,1\}^m$ which is not in $\bigcup_i (S \oplus z_i)$ is at most $\frac{2^m}{m^m}$.

We now prove the theorem. Given $L \in \mathcal{BPP}$, there exist a polynomial m and an algorithm M such that M uses $m(|x|)$ random coins and errs with probability less than $1/m$. For any input x, let $S_x \subseteq \{0,1\}^{m(|x|)}$ denote the set of random coins for which $M(x;r)$ outputs 1. Thus, if $x \in L$ (letting $m = m(|x|)$) we have $|S_x| > (1 - \frac{1}{m}) \cdot 2^m$ while if $x \notin L$ then $|S_x| < \frac{2^m}{m}$. This leads to the following Σ_2 characterization of L:

$$x \in L \Leftrightarrow \exists z_1, \ldots, z_m \in \{0,1\}^m \; \forall y \in \{0,1\}^m : y \in \bigcup_i (S_x \oplus z_i).$$

(Note the desired condition can be efficiently verified by checking if $M(x; y \oplus z_i) \stackrel{?}{=} 1$ for some i.) ■

2 Interactive Proofs

Let us begin by re-examining our intuitive notion of what it means to "prove" a statement. Traditional mathematical proofs are *static* and are verified *deterministically*: the verifier checks the claimed proof of a given statement and is either convinced that the statement is true (if the proof is correct) or remains unconvinced (if the proof is flawed — note that the statement may possibly still be true in this case, it just means there was something wrong with the proof). A statement is true (in this traditional setting) iff there *exists* a valid proof that convinces a legitimate verifier.

Abstracting this process a bit, we may imagine a prover $\mathbf{P}$ and a verifier $\mathbf{V}$ such that the prover is trying to convince the verifier of the truth of some particular statement x; more concretely, let us say that $\mathbf{P}$ is trying to convince $\mathbf{V}$ that $x \in L$ for some fixed language L. We will require the verifier to run in polynomial time (in $|x|$), since we would like whatever proofs we come up with to be efficiently verifiable. A traditional mathematical proof can be cast in this framework by simply having $\mathbf{P}$ send a proof π to $\mathbf{V}$, who then deterministically checks whether π is a valid proof of x and outputs $\mathbf{V}(x, \pi)$ (with 1 denoting acceptance and 0 rejection). (Note that since $\mathbf{V}$ runs in polynomial time, we may assume that the length of the proof π is also polynomial.) The traditional mathematical notion of a proof is captured by requiring:

- If $x \in L$, then there *exists* a proof π such that $\mathbf{V}(x, \pi) = 1$.
- If $x \notin L$, then *no matter what proof* π the prover sends we have $\mathbf{V}(x, \pi) = 0$.

We refer to the above as a type of proof system, a term we will define more formally later. It should be obvious that L has a proof system of the above sort iff $L \in \mathcal{NP}$.

There are two ways the above can be generalized. First, we can allow the verifier to be *probabilistic*. Assume for a moment that we restrict the prover to sending an empty proof. If the verifier is deterministic, then a language L has a proof system of this sort only if $L \in \mathcal{P}$ (as the prover is no help here). But if the verifier is probabilistic then we can handle any $L \in \mathcal{BPP}$ (if we allow two-sided error). If we go back to allowing non-empty proofs, then we already gain something: we can eliminate the error when $x \in L$. To see this, recall the proof that $\mathcal{BPP} \in \Sigma_2$. The basic idea was that if a set $S \subset \{0,1\}^\ell$ is "small" then for any strings $z_1, \ldots, z_\ell \in \{0,1\}^\ell$, the set $\bigcup_i (S \oplus z_i)$ is still "small." To make this concrete, say $|S| \leq 2^\ell/4\ell$. Then for any $z_1, \ldots, z_\ell$ we have:

$$\left| \bigcup_{i=1}^{\ell} (S \oplus z_i) \right| \leq \ell \cdot |S| \leq 2^\ell/4. \tag{1}$$

On the other hand, if S is "large" (specifically, if $|S| \geq (1 - \frac{1}{4\ell}) \cdot 2^\ell$) then there exist $z_1, \ldots, z_m$ such that $\bigcup_i (S \oplus z_i) = \{0,1\}^\ell$.

The above leads to the following proof system for any $L \in \mathcal{BPP}$: Let M be a $\mathcal{BPP}$ algorithm deciding L, using a random tape of length ℓ, and having error at most $1/4\ell$ (for some polynomial ℓ). The prover sends a proof $\pi = (z_1, \ldots, z_\ell)$ to the verifier (where each $z_i \in \{0,1\}^\ell$); $\mathbf{V}$ then chooses a random $r \in \{0,1\}^\ell$ and accepts iff

$$\bigvee_{i=1}^{\ell} M(x; r \oplus z_i) = 1.$$

For common input x, let S_x be the set of random coins for which $M(x) = 1$. If $x \in L$, then $|S_x| \geq (1 - \frac{1}{4\ell}) \cdot 2^\ell$ and so there does indeed exist $\pi = (z_1, \ldots, z_\ell)$ such that $r \in \bigcup_i (S_x \oplus z_i)$ for *every* $r \in \{0,1\}^\ell$. Fixing such a π, this means that for every r there exists an index i for which $r \in S_x \oplus z_i$, and so $r \oplus z_i \in S_x$. Thus, if the prover sends this π the verifier will always accept. On the other hand, if $x \notin L$ then $|S_x| \leq 2^\ell/4\ell$ and so, using Eq. (**??**), we have

$$\Pr_{r \in \{0,1\}^\ell} \left[r \in \bigcup_{i=1}^{\ell} (S \oplus z_i) \right] \leq 1/4.$$

So $\mathbf{V}$ accepts in this case with probability at most $1/4$.

To summarize, we have shown a proof system for any $L \in \mathcal{BPP}$ such that:

- If $x \in L$, then there *exists* a proof π such that $\Pr[\mathbf{V}(x, \pi) = 1] = 1$.
- If $x \notin L$, then *no matter what proof* π the prover sends we have $\Pr[\mathbf{V}(x, \pi) = 1] \leq 1/4$.

Thus, assuming $\mathcal{P} \neq \mathcal{BPP}$, we see that *randomization* helps. And assuming $\mathsf{co}\mathcal{RP} \neq \mathcal{BPP}$, allowing *communication from the prover to the verifier* helps.

We can further generalize proof systems by allowing *interaction* between the prover and verifier. (One can think of this as allowing the verifier to ask questions. In this sense, the notion of a proof becomes more like a lecture than a static proof written in a book.) Note that unless we also allow randomness, allowing interaction will not buy us anything: if the verifier is deterministic then the

prover can predict all the verifier's questions in advance, and simply include all the corresponding answers as part of the (static) proof.

Before we explore the additional power of interaction, we introduce some formal definitions. For interactive algorithms $\mathbf{P}, \mathbf{V}$, let $\langle \mathbf{P}, \mathbf{V} \rangle (x)$ denote the output of $\mathbf{V}$ following an interaction of $\mathbf{P}$ with $\mathbf{V}$ on common input x.

Definition 1 *$L \in \mathcal{IP}$ if there exist a pair of interactive algorithms $(\mathbf{P}, \mathbf{V})$, with $\mathbf{V}$ running in probabilistic polynomial time (in the length of the common input x), such that*

1. *If $x \in L$, then $\Pr[\langle \mathbf{P}, \mathbf{V} \rangle (x) = 1] = 1$.*
2. *If $x \notin L$, then for* any *(even cheating) $\mathbf{P}^*$ we have $\Pr[\langle \mathbf{P}^*, \mathbf{V} \rangle (x) = 1] \leq 1/2$.*

(We stress that $\mathbf{P}$ and $\mathbf{P}^*$ are allowed to be computationally unbounded.) $(\mathbf{P}, \mathbf{V})$ satisfying the above are called a proof system *for L. We say $L \in \mathcal{IP}[\ell]$ if it has a proof system as above using $\ell = \ell(|x|)$ rounds of interaction (where each message sent by either party counts as a round).*

Using this notation, we have seen already that $\mathcal{NP} \cup \mathcal{BPP} \subseteq \mathcal{IP}[1]$.

Some comments on the definition are in order:

- One could relax the definition to allow for *two-sided* error, i.e., error even when $x \in L$. It is known, however, that this results in an equivalent definition [**?**] (although the round complexity increases by a constant). On the other hand, if the definition is "flipped" so that we allow error only when $x \in L$ (and require no error when $x \notin L$) we get a definition that is equivalent to $\mathcal{NP}$.
- As usual, the error probability of $1/2$ is arbitrary, and can be made exponentially small by repeating the proof system suitably many times. (It is easy to see that sequential repetition works, and a more detailed proof shows that parallel repetition works also [**?**, Appendix C].)
- Although the honest prover is allowed to be computationally unbounded, it suffices for it to be a PSPACE machine. In certain cases it may be possible to have $\mathbf{P}$ run in polynomial time (for example, if $L \in \mathcal{NP}$ and $\mathbf{P}$ is given a proof π as auxiliary information). In general, it remains an open question as to how powerful $\mathbf{P}$ needs to be in order to give a proof for some particular class of languages.[1]

2.1 Graph Non-Isomorphism is in $\mathcal{IP}$

It is possible to show that $\mathcal{IP} \subseteq$ PSPACE (since, fixing some $\mathbf{V}$ and some x, we can compute the optimal prover strategy in polynomial space). But does interaction buy us anything? Does $\mathcal{IP}$ contain anything more than $\mathcal{NP}$ and $\mathcal{BPP}$? We begin by showing the rather surprising result that graph *non*-isomorphism is in $\mathcal{IP}$.

If G is an n-vertex graph and π is a permutation on n elements, we let $\pi(G)$ be the n-vertex graph in which

$$(i, j) \text{ is an edge in } G \Leftrightarrow (\pi(i), \pi(j)) \text{ is an edge in } \pi(G).$$

Note that G_0 is isomorphic to G_1 (written $G_0 \cong G_1$) iff $G_0 = \pi(G_1)$ for some π. (We identify a graph with its adjacency matrix. So, there is a difference between two graphs being *equal* [i.e., having the *same* adjacency matrix] and being *isomorphic*.)

Let G_0, G_1 be two graphs. The proof system for graph non-isomorphism works as follows:

[1] For example, we will soon see that $\mathsf{co}\mathcal{NP} \subseteq \mathcal{IP}$. By what we have just said, we know that if $L \in \mathsf{co}\mathcal{NP}$ then there exists a proof system for L with a prover running in PSPACE. But we do not know whether there exists a proof system for L with a prover running in, say, $\mathcal{P}^{\mathsf{co}\mathcal{NP}} = \mathcal{P}^{\mathcal{NP}}$.

1. The verifier chooses a random bit b and a random permutation π, and sends $G' = \pi(G_b)$.
2. If $G' \cong G_0$, the prover replies with 0; if $G' \cong G_1$, it replies with 1.
3. The verifier accepts iff the prover replies with $\mathbf{V}$'s original bit b.

Note that if $G_0 \ncong G_1$, then it cannot be the case that both of $G' \cong G_0$ and $G' \cong G_1$ hold; so, the prover always answers correctly. On the other hand, if $G_0 \cong G_1$ (so that (G_0, G_1) is *not* in the language) then the verifier's bit b is completely hidden to the prover (even though the prover is all-powerful!); this is because a random permuted copy of G_0 is in this case distributed identically to a random permuted copy of G_1. So when G_0, G_1 *are* isomorphic, even a cheating prover can only make the verifier accept with probability $1/2$.

3 Public-Coin Proof Systems

Crucial to the above protocol for graph non-isomorphism is that the verifier's coins are *private*, i.e., hidden from the prover. At around the same time the class $\mathcal{IP}$ was proposed, a related class was proposed in which the verifier's coins are required to be *public* (still, the verifier does not toss coins until they are needed, so that the prover does not know what coins will be tossed in the future). These are called *Arthur-Merlin* proof systems, where Arthur represents the (polynomial-time) verifier and Merlin the (all-powerful) prover. We again require perfect completeness and bounded soundness (though see Theorems **??** and **??** below). As in the case of $\mathcal{IP}$ one can in general allow polynomially many rounds of interaction. Although it might appear that Arthur-Merlin proofs are (strictly) *weaker* than general interactive proofs, this is not the case [**?**]. We do not prove this, but an indication of the general technique will be given in Section **??**.

We will consider for now only the Arthur-Merlin classes **MA** and **AM** where there are one or two rounds of interaction. For the class **MA** Merlin talks first, and then Arthur chooses random coins and tries to verify the "proof" that Merlin sent. (We have already seen this type of proof system before when we showed an interactive proof for $\mathcal{BPP}$.) For the class **AM** Arthur talks first but is limited to sending its random coins (so the previous proof of graph non-isomorphism does not satisfy this); then Merlin sends a proof that is supposed to "correspond" to these random coins, and Arthur verifies it. (Arthur does not choose any additional random coins after receiving Merlin's message, although it would not change the class if Arthur did; see Theorem **??**, below.) One can also express these in the following definition, which is just a specialization of the general definition of Arthur-Merlin proofs to the above cases:

Definition 2 *$L \in \mathbf{MA}$ if there exists a deterministic algorithm $\mathbf{V}$ running in polynomial time (in the length of its first input) such that:*

- *If $x \in L$ then $\exists \pi$ such that for all r we have $\mathbf{V}(x, \pi, r) = 1$.*
- *If $x \notin L$ then $\forall \pi$ we have $\Pr_r[\mathbf{V}(x, \pi, r) = 1] \leq 1/2$.*

$L \in \mathbf{AM}$ if there exists a deterministic algorithm $\mathbf{V}$ running in polynomial time (in the length of its first input) such that:

- *If $x \in L$ then for all r there exists a π such that $\mathbf{V}(x, r, \pi) = 1$.*
- *If $x \notin L$ then $\Pr_r[\exists \pi : \mathbf{V}(x, r, \pi) = 1] \leq 1/2$.*

In the case of **MA** *the prover (Merlin) sends π and the verifier (Arthur) then chooses random coins r, while in the case of* **AM** *the verifier (Arthur) sends random coins r and then the prover (Merlin) responds with π.*

MA can be viewed as a randomized version of $\mathcal{NP}$ (since a fixed proof is verified using random-

ization) and so a language in **MA** is sometimes said to have "publishable proofs." It is clear that Arthur-Merlin proofs are not more powerful than the class $\mathcal{IP}$ (since an Arthur-Merlin proof system is a particular kind of proof system).

Theorem 5 $\mathbf{MA} \subseteq \mathbf{AM}$.

Proof Say $L \in \mathbf{MA}$. Then we have an **MA** proof system with perfect completeness and soundness error at most $1/2$. Say the message π sent by Merlin has length $p(|x|)$ for some polynomial p. Using error reduction, we can obtain a proof system with perfect completeness and soundness error at most $1/2^{p+1}$; note that the lengths of the messages sent by Merlin do not change (only the lengths of the random coins r used by Arthur increase). So, when $x \in L$ there exists a π (call it π^*) for which $\mathbf{V}(x, \pi^*, r) = 1$ for all r chosen by Arthur, while if $x \notin L$ then for any π sent by Merlin the fraction of r for which Arthur accepts is at most $1/2^{p+1}$. Now simply flip the order of messages: first Arthur will choose r and send it to Merlin, and then Merlin replies with a π and Arthur verifies exactly as before. If $x \in L$ then Merlin has no problem, and can simply send π^*. On the other hand, if $x \notin L$ then what is the probability that there *exists* a π that will cause Arthur to accept? Well, for any *fixed* π the probability that π will work is at most $1/2^{p+1}$. Taking a union bound over *all* π, we see that the probability that there exists one that works is at most $1/2$. We conclude that $L \in \mathbf{AM}$. ■

As we have said, the same proof shows that an "**MA**" step can be replaced by an "**AM**" step in general. So, $\mathbf{AMA} = \mathbf{AAM} = \mathbf{AM}$ and[2] $\mathbf{MAM} = \mathbf{AMM} = \mathbf{AM}$, and so on. In fact, the above proof technique shows that any Arthur-Merlin proof system with a *constant* number of rounds collapses to exactly **AM** (except for **MA** which may be strictly contained in **AM**). Note that the proof does not extend to proof systems with an arbitrary (non-constant) number of rounds since the communication complexity increases by a multiplicative factor each time an "**MA**" step is replaced by an "**AM**" step (and so if we perform this switch too many times, the communication will no longer be polynomial).

The classes **MA** and **AM** do not change if we allow error when $x \in L$. We now prove this. Let $\mathbf{MA}_\varepsilon$ and $\mathbf{AM}_\varepsilon$ denote the corresponding classes when (bounded) two-sided error is allowed.

3.1 Evidence that Graph Isomorphism is not $\mathcal{NP}$-Complete

Let GI be the language of graph isomorphism, and GNI be the language of graph non-isomorphism. In the previous section we showed $GNI \in \mathbf{AM}$. This gives evidence that GI is *not* $\mathcal{NP}$-complete.

Theorem 6 *If GI is $\mathcal{NP}$-complete, then the polynomial hierarchy collapses (specifically, $\mathsf{PH} = \Sigma_2$).*

Proof We first observe that $\mathbf{AM} \subseteq \Pi_2$ (why?). Now, assume GI is $\mathcal{NP}$-complete. Then GNI is $\mathsf{co}\mathcal{NP}$-complete and hence (since $GNI \in \mathbf{AM}$) we have $\mathsf{co}\mathcal{NP} \subseteq \mathbf{AM}$. We show that this implies $\Sigma_2 \subseteq \mathbf{AM} \subseteq \Pi_2$ and hence $\mathsf{PH} = \Sigma_2$.

Say $L \in \Sigma_2$. Then by definition of Σ_2, there is a language $L' \in \Pi_1 = \mathsf{co}\mathcal{NP}$ such that: (1) if $x \in L$ then there exists a y such that $(x, y) \in L'$, but (2) if $x \notin L$ then for all y we have $(x, y) \notin L'$. This immediately suggests the following proof system for L:

1. Merlin sends y to Arthur.

[2] The theorem shows that $\mathbf{AMA} \subseteq \mathbf{AAM} = \mathbf{AM}$, but the inclusion $\mathbf{AM} \subseteq \mathbf{AMA}$ is trivial.

2. Arthur and Merlin then run an **AM** protocol that $(x, y) \in L'$ (this is possible precisely because $L' \in \mathsf{co}\mathcal{NP} \subseteq \mathbf{AM}$).

The above is an **MAM** proof system for L. But, as we have seen, this means there is an **AM** proof system for L. Since $L \in \Sigma_2$ was arbitrary this means $\Sigma_2 \subseteq \mathbf{AM}$, completing the proof. ■

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] Wikipedia: https://en.wikipedia.org/wiki/Chernoff_bound