Notes on Complexity Theory: Fall 2005 Last updated: November, 2005

Lecture 10

Jonathan Katz

1 Interactive Proofs

1.1 Introduction and Motivation

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 in a *deterministic* way: the reader (or "verifier") checks the claimed proof of a given statement and is thereby 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 $\mathcal{P}$ and a verifier $\mathcal{V}$ such that the prover is trying to convince the verifier of the truth of some particular statement x; more concretely, let us say that $\mathcal{P}$ is trying to convince $\mathcal{V}$ 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. Then a traditional mathematical proof can be cast in this framework by simply having $\mathcal{P}$ send a proof π to $\mathcal{V}$, who then deterministically checks whether π is a valid proof of x and outputs $\mathcal{V}(x, \pi)$ (with 1 denoting acceptance and 0 rejection). (Note that since $\mathcal{V}$ runs in polynomial time, we may assume that the length of the proof π is also polynomial.) Now, the traditional mathematical notion of a proof is captured by requiring the following:

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

We refer to $(\mathcal{P}, \mathcal{V})$ satisfying the above as a *proof system for L* (we will define this more formally later). It should be obvious at this point that L has a proof system of the above sort iff $L \in \mathcal{NP}$.[1]

Before we generalize the above, let us first *restrict* the above by requiring π to be the empty string (in other words, the prover is doing nothing). Now, L has a proof system of this sort (i.e., one in which the prover does not communicate with the verifier) iff $L \in \mathbf{P}$.[2]

As one might expect at this point in the course, one way to generalize the above notion of a proof system is to allow the verifier to be *probabilistic*, and to introduce the possibility of error. Let us first see what happens if we try this generalization when π is empty. Now, we require:

- If $x \in L$, then the probability (over random coins of $\mathcal{V}$) that $\mathcal{V}(x) = 1$ is at least 3/4.
- If $x \notin L$, then $\Pr[\mathcal{V}(x) = 1] \leq 1/4$.

Note that L has such a proof system iff $L \in \mathcal{BPP}$.

[1] We do not claim, however, that the language of provably-true mathematical statements is in $\mathcal{NP}$; this was for motivation only.

[2] For this lecture, we denote the class of languages decidable in polynomial time by $\mathbf{P}$ (since $\mathcal{P}$ is now our prover).

What if we combine this idea of allowing $\mathcal{V}$ to be probabilistic with the possibility of having $\mathcal{P}$ send a *non*-empty proof to $\mathcal{V}$? Now things get more interesting. Let $L \in \mathcal{BPP}$. We may first observe that, once we allow non-empty proofs, *we can eliminate the error when* $x \in L$. To see this, let us recall Lautemann's proof (cf. Lecture 7) 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:

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

(the parameters here are a little different from what we used in Lecture 7). On the other hand, if S is "large" (specifically, if $|S| \geq (1 - \frac{1}{4\ell}) \cdot 2^\ell$) then using the probabilistic method we can show that there exist $z_1, \ldots, z_m$ such that $\bigcup_i (S \oplus z_i) = \{0,1\}^\ell$ (see the proof in Lecture 7).

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 $z_i \in \{0,1\}^\ell$); $\mathcal{V}$ 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. (1), we have

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

So $\mathcal{V}$ 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 $\Pr[\mathcal{V}(x, \pi) = 1] = 1$.
- If $x \notin L$, then *no matter what proof* π the prover sends we have $\Pr[\mathcal{V}(x, \pi) = 1] \leq 1/4$.

Assuming $\mathbf{P} \neq \mathcal{BPP}$, we see that *randomization* helps. Assuming co$\mathcal{RP} \neq \mathcal{BPP}$, we see that allowing *communication from the prover to the verifier* helps (since we can achieve perfect completeness). What if we generalize this notion of communication to allow *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, it is time to introduce some formal definitions.

1.2 Definitions

We define the class $\mathcal{IP}$ of languages admitting *interactive proofs* [5]. For interactive algorithms $\mathcal{P}, \mathcal{V}$, let $\langle \mathcal{P}, \mathcal{V} \rangle(x)$ denote the output of $\mathcal{V}$ following an interaction of $\mathcal{P}$ with $\mathcal{V}$ on common input x.

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

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

$(\mathcal{P}, \mathcal{V})$ satisfying the above are called a *proof system* for L. We stress that $\mathcal{P}$ and $\mathcal{P}^*$ are allowed to be computationally unbounded.

We let $\mathcal{IP}[n]$ denote a proof system using $n = n(|x|)$ rounds of interaction (where each message sent by either party counts as a round). Note that we may always assume that the prover sends the last message. Also, n must be polynomial since $\mathcal{V}$ runs in polynomial time. $\diamondsuit$

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 a small probability of error even when $x \in L$. It is known, however, that this results in an equivalent definition [3] (however, the efficiency of the proof system [e.g., round complexity] in each case may[3] be different). 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 [4, Appendix C].)
- Although $\mathcal{P}$ is allowed to be computationally unbounded, in fact it "only" needs to be a PSPACE machine. Note also that in certain cases it may be possible to have $\mathcal{P}$ run in polynomial time (for example, if $L \in \mathcal{NP}$ and $\mathcal{P}$ is given a proof π as auxiliary information). In general, it remains an open question as to how powerful $\mathcal{P}$ needs to be in order to give a proof of some particular class of languages.[4]

1.3 On the Power of $\mathcal{IP}$

It is not too difficult to see that $\mathcal{IP} \subseteq$ PSPACE (since we can compute the optimal prover strategy in polynomial space). But does interaction buy us anything? Can we show that $\mathcal{IP}$ contains anything more than $\mathcal{NP}$ and $\mathcal{BPP}$? We show the rather surprising result that graph *non*-isomorphism is in $\mathcal{IP}$. First, some notation: 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

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

[3]Thus, if one is concerned with questions of efficiency it is important to be clear what definition is being considered.

[4]For example, we will soon see that co$\mathcal{NP} \subseteq \mathcal{IP}$. By what we have just said, we know that if $L \in$ 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, $\mathbf{P}^{\mathrm{co}\mathcal{NP}} = \mathbf{P}^{\mathcal{NP}}$.

Note that G_0 is isomorphic to G_1 (written $G_0 \cong G_1$) iff $G_0 = \pi(G_1)$ for some π. (For the present discussion 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. 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 $\mathcal{V}$'s original bit b.

Note that if $G_0 \not\cong G_1$, 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 then 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.

1.4 On Public vs. Private Coins

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 those coins will be ahead of time) [1, 2]. These are called *Arthur-Merlin* proof systems, where Arthur represents the (poly-time) verifier and Merlin the (all-powerful) prover. We again require perfect completeness and bounded soundness (though see Theorems 2 and 3 below). As in the case of $\mathcal{IP}$ one can in general allow arbitrary (polynomial) rounds of interaction, but we will consider for now only the classes **MA** and **AM**. 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 $\mathcal{BPP}$ in the Introduction.) For the class **AM** Arthur talks first but is limited to sending its random coins only (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 4, 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 $\mathcal{V}$ running in polynomial time (in the length of its first input) such that:

- If $x \in L$ then $\exists y$ such that for all z (of some fixed polynomial length) we have $\mathcal{V}(x, y, z) = 1$.
- If $x \notin L$ then $\forall y$ we have $\Pr_z[\mathcal{V}(x, y, z) = 1] \leq 1/2$.

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

- If $x \in L$ then for all y (of some fixed polynomial length) there exists a z such that $\mathcal{V}(x, y, z) = 1$.
- If $x \notin L$ then $\Pr_y[\exists z : \mathcal{V}(x, y, z) = 1] \leq 1/2$.

Note that in the case of **MA** the prover (Merlin) is sending y and the verifier (Arthur) then chooses z at random, while in the case of **AM** the verifier (Arthur) sends a random y and then the prover (Merlin) responds with z. $\diamondsuit$

MA can be viewed as a randomized version of $\mathcal{NP}$ (since a fixed proof is verified using randomization) 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). Although it might appear that Arthur-Merlin proofs are (strictly) *weaker* than general interactive proofs, this is not the case. In fact:

Theorem 1 ([6]) *$\mathcal{IP}[n] \subseteq$ **AM**$[n+2]$ for any polynomial n.*

We do not prove the above, but an indication of the general technique will be given in Section 1.5.

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

Theorem 2 $\mathbf{AM}_\varepsilon = \mathbf{AM}$.

Proof Say $L \in \mathbf{AM}_\varepsilon$. Using standard error reduction, we thus have a proof system for L in which Arthur sends a random string y of (polynomial) length ℓ and the error is less than $1/4\ell$. For a common input x, let S_x denote the set of challenges y (that Arthur can send) for which there exists a z (that Merlin can send) such that $\mathcal{V}(x, y, z) = 1$ (i.e., Arthur will accept). We know by definition of $\mathbf{AM}_\varepsilon$ that if $x \in L$ then $|S_x| \geq (1 - \frac{1}{4\ell}) \cdot 2^\ell$ while if $x \notin L$ then $|S_x| \leq 2^\ell/4\ell$. Exactly as in the proof system for $\mathcal{BPP}$ in the Introduction, this means that we have the following proof system for L:

1. Merlin sends $(a_1, \ldots, a_\ell)$, each of which is an ℓ-bit string.
2. Arthur sends random $r \in \{0,1\}^\ell$.
3. Merlin proves that $r \in \bigcup_i (S_x \oplus a_i)$ by finding an i such that $r \oplus a_i \in S_x$, setting $y = r \oplus a_i$, and then computing the appropriate response z to the "challenge" y. So, Merlin's response will consist of i and this z.
4. Arthur runs $\mathcal{V}(x, r \oplus a_i, z)$ and outputs the result.

The above has perfect completeness and soundness error at most $1/4$ (we do not go through the analysis since it is the same as in the $\mathcal{BPP}$ case).

The problem is that the above is a three-round proof system (notationally, it is an **MAM** proof system)! But we will show in Theorem 4 that an "**MA**" step can be replaced by an "**AM**" step (while preserving perfect completeness), and so if we apply that here and then combine Merlin's last two messages we get an **AMM** = **AM** protocol. $\blacksquare$

Theorem 3 $\mathbf{MA}_\varepsilon = \mathbf{MA}$.

Proof Things are a bit easier here. Let $L \in \mathbf{MA}_\varepsilon$. Now we know that there is a proof system such that if $x \in L$ then there exists a y (that Merlin can send) that will cause Arthur to accept with high probability (i.e., $\mathcal{V}(x, y, z) = 1$ with high probability over choice of z), while if $x \notin L$ then for any y (that Merlin sends), Arthur will accept only with low probability (i.e., $\mathcal{V}(x, y, z) = 1$ with low probability over choice of z). For a given x *and* y, let $S_{x,y}$ denote the set of coins z for which $\mathcal{V}(x, y, z) = 1$. So if $x \in L$ there exists a y for which $S_{x,y}$ is "large," while if $x \notin L$ then for *every* y the set $S_{x,y}$ is "small." Having Merlin send y along with a proof π that $S_{x,y}$ is "large" (exactly as in the $\mathcal{BPP}$ case) gives the desired result. ■

As promised in the first proof above, we now show that **MA** $\subseteq$ **AM**. More generally, the proof technique extends to show that an "**MA**" step can be replaced by an "**AM**" step.

Theorem 4 MA $\subseteq$ AM.

Proof Say $L \in$ **MA**. Then we have an **MA** proof system with perfect completeness and soundness error at most 1/2. Say the message y 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 z used by Arthur increase). So, when $x \in L$ there exists a y (call it y^*) for which $\mathcal{V}(x, y^*, z) = 1$ for all z chosen by Arthur, while if $x \notin L$ then for any y sent by Merlin the fraction of z for which Arthur will accept is at most $1/2^{p+1}$. Now simply flip the order of messages: first Arthur will choose z and send it to Merlin, and then Merlin replies with a y and Arthur verifies exactly as before. If $x \in L$ then Merlin has no problem, and can simply send y^*. On the other hand, if $x \notin L$ then what is the probability that there exists a y that will cause Arthur to accept? Well, for any *fixed* y the probability that y will work is at most $1/2^{p+1}$. Taking a union bound over *all* y, we see that the probability that there exists one that works is at most 1/2. We conclude that $L \in$ **AM**. ■

As we have said above, the same proof shows that an "**MA**" step can be replaced by an "**AM**" step in general. So, **AMA** = **AAM** = **AM** and[5] **MAM** = **AMM** = **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 $\omega(1)$ rounds since the communication complexity blows up 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). See [1, 2] for more on "speedup" theorems of this type.

1.5 Graph Non-Isomorphism in AM

The proof system we showed earlier for graph non-isomorphism relied on the fact that the verifier's coins are kept hidden from the prover. Is this inherent? Somewhat surprisingly, we now show a *public-coin* proof for graph non-isomorphism. We begin by introducing some more notation. For an n-vertex graph G (again viewed as an adjacency matrix), consider the (multi-)set $\mathsf{all}(G) = \{\pi_1(G), \ldots, \pi_{n!}(G)\}$ of all permuted versions of G. This is indeed a multi-set (in general) since it is possible that $\pi_i(G) = \pi_j(G)$ even when $\pi_i \neq \pi_j$. For example, consider the 3-vertex graph G in which there is a single edge $(1, 2)$. Considering the 6 possible permutations on the labels of the vertices, we see that $\pi = (12)(3)$ maps G to itself, even though π is not the identity permutation. On the other hand, $\pi' = (13)(2)$ maps G to a graph isomorphic to, but not identical to, G.

Let $\mathsf{aut}(G) = \{\pi \mid \pi(G) = G\}$; these are the *automorphisms* of G. Note that $\mathsf{aut}(G)$ is never empty, since the identity permutation is always in $\mathsf{aut}(G)$. Let $\mathsf{iso}(G)$ be the *set* (not multi-set) $\{\pi(G) \mid \pi \text{ is a permutation}\}$. We claim that for any n-vertex graph G we have:

$$|\mathsf{aut}(G)| \cdot |\mathsf{iso}(G)| = n! \ .$$

[5]The theorem shows that **AMA** $\subseteq$ **AAM** = **AM**, but the inclusion **AM** $\subseteq$ **AMA** is trivial.

The reason is that our original multi-set $\mathsf{all}(G)$ has exactly $n!$ elements in it, but each graph in $\mathsf{iso}(G)$ appears exactly $\mathsf{aut}(G)$ times in $\mathsf{all}(G)$ (because $|\mathsf{aut}(G)| = |\mathsf{aut}(\pi(G))|$ for any permutation π).

We now have the ideas we need to describe the proof system. Given graphs (G_0, G_1), define the set W as follows:

$$W = \left\{ (H, \sigma) \mid \begin{array}{c} H \text{ is isomorphic to either } G_0 \text{ or } G_1 \\ \text{and } \sigma \in \mathsf{aut}(H) \end{array} \right\}.$$

Note that if $G_0 \cong G_1$, then H is isomorphic to G_0 iff it is isomorphic to G_1; also, the number of automorphisms of any such H is exactly $|\mathsf{aut}(G_0)|$. So the size of W is exactly $|\mathsf{iso}(G_0)| \cdot |\mathsf{aut}(G_0)| = n!$. On the other hand, if $G_0 \not\cong G_1$ then the graphs isomorphic to G_0 are distinct from those graphs isomorphic to G_1. So the size of W in this case is

$$|\mathsf{iso}(G_0)| \cdot |\mathsf{aut}(G_0)| + |\mathsf{iso}(G_1)| \cdot |\mathsf{aut}(G_1)| = 2n!\ .$$

So, $|W \times W| = (n!)^2$ if $G_0 \cong G_1$ and $|W \times W| = 4 \cdot (n!)^2$ if $G_0 \not\cong G_1$. Furthermore, $W \in \mathcal{NP}$ since we can prove membership in W by giving an isomorphism to either G_0 or G_1 (and the automorphism can be verified in polynomial time). This suggests the following proof system:

1. On common input (G_0, G_1), define $W \times W$ as above. (The verifier obviously cannot construct $W \times W$, but all it needs to do is compute the upper bound $4(n!)^2$ on its size.) Let $m = \log 4(n!)^2$, and note that m is polynomial in the input size n.

2. Arthur selects a random hash function h from a pairwise-independent hash family mapping strings of the appropriate length (which will becomes obvious in a minute) to strings of length m. It sends h to Merlin.

3. Merlin finds an $x \in W \times W$ such that $h(x) = 0^m$ (if one exists). It sends this x to Arthur, along with an $\mathcal{NP}$-proof that $x \in W \times W$.

4. Arthur verifies the message it receives in the obvious way.

Say (G_0, G_1) are isomorphic. Then $|W \times W| = (n!)^2$ and so

$$\begin{aligned} \Pr_h[\exists x \in W \times W : h(x) = 0^m] &\leq \sum_{x \in W \times W} \Pr_h[h(x) = 0^m] \\ &= (n!)^2 \cdot 2^{-m} \ = \ 1/4, \end{aligned}$$

and so Merlin convinces Arthur only with probability at most 1/4. On the other hand, if $G_0 \not\cong G_1$ then $|W \times W| = 4(n!)^2$ and we can bound the desired probability as follows:

$$\begin{aligned} \Pr_h[\exists x \in W \times W : h(x) = 0^m] &\geq \sum_{x \in W \times W} \Pr_h[h(x) = 0^m] \\ &\quad - \sum_{\substack{x,y \in W \times W \\ x \neq y}} \Pr_h[h(x) = 0^m \wedge h(y) = 0^m] \\ &> 1 - \frac{(4(n!)^2)^2}{2} \cdot (2^{-m})^2 \ = \ \frac{1}{2}, \end{aligned}$$

using the inclusion-exclusion principle for the first inequality, and relying on pairwise independence in the second step (a better bound can be obtained using Chebyshev's inequality).

The above does not have perfect completeness, but we can remedy this using Theorem 2.

1.6 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 $GNI \in \mathbf{AM}$. We now show that this gives evidence that GI is *not* $\mathcal{NP}$-complete. (Recall that if $\mathbf{P} \neq \mathcal{NP}$ then there exist problems in $\mathcal{NP} \setminus \mathbf{P}$ that are not $\mathcal{NP}$-complete (by Ladner's theorem). But GI would be a "natural" example of such a language.)

Theorem 5 *If GI is $\mathcal{NP}$-complete, then the polynomial hierarchy collapses (specifically, $\mathsf{PH} = \Sigma_2$).*

Proof If GI is $\mathcal{NP}$-complete, then GNI is co$\mathcal{NP}$-complete. Since $GNI \in \mathbf{AM}$ this would imply co$\mathcal{NP} \subseteq \mathbf{AM}$. We show that this implies the collapse of PH to the second level.

It is an easy observation that $\mathbf{AM} \subseteq \Pi_2$ (why?). We show that $\Sigma_2 \subseteq \mathbf{AM}$ which implies $\Sigma_2 = \Pi_2$ and hence $\mathsf{PH} = \Sigma_2$ (see Lecture 6).

Say $L \in \Sigma_2$. Then, by definition of Σ_2, there is a language $L' \in \Pi_1 = $ 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. Arthur and Merlin then run an $\mathbf{AM}$ protocol that $(x, y) \in L'$ (this is possible precisely because $L' \in$ co$\mathcal{NP} \subseteq \mathbf{AM}$).

The above is an $\mathbf{MAM}$ proof system for L. But by Theorem 4, this means there is an $\mathbf{AM}$ proof system for L. Since $L \in \Sigma_2$ was arbitrary, this means $\Sigma_2 \subseteq \mathbf{AM}$. This completes the proof. ■

References

[1] L. Babai. Trading Group Theory for Randomness. STOC '85.

[2] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *J. Computer and System Sciences* 36(2): 254–276, 1988.

[3] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. In *Advances in Computing Research: A Research Annual*, vol. 5 (Randomness and Computation, S. Micali, ed.), 1989. Available at http://www.wisdom.weizmann.ac.il/˜oded/papers.html

[4] O. Goldreich. *Modern Cryptography, Probabilistic Proofs, and Pseudorandomness.* Springer-Verlag, 1998.

[5] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM. J. Computing* 18(1): 186–208, 1989.

[6] S. Goldwasser and M. Sipser. Private Coins vs. Public Coins in Interactive Proof Systems. STOC '86.