CS 710: Complexity Theory 4/20/2010

Lecture 24: Interactive Proofs

Instructor: Dieter van Melkebeek Scribe: Jeff Kinne and Ashutosh Kumar

Last time we introduced the notion of universal families of hash functions and used them to establish results that relate the counting complexity classes to the polynomial hierarchy. We were in the middle of the proof of the theorem stating that any language in the polynomial hierarchy can be decided in polynomial time with a single query to a #P function. Today we'll complete this proof by completing the proofs of the inclusions $\mathrm{PH} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$ and $\mathrm{BPP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$.

Next we'll introduce the notion of interactive proof systems which models computation as a result of interaction between two parties. We make some remarks on how various modifications of the definition affect the class of languages this model can compute. We'll define public/private coin protocols, and first show that the graph non-isomorphism problem admits an interactive proof using private coins. After that, we'll also show that there is an interactive proof for the same problem that uses *public* coins. In subsequent lectures, we'll see that this provides further evidence that the graph isomorphism problem is not NP-complete.

1 Proof of $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$

Note that we had already shown that $\mathrm{NP} \subseteq \mathrm{RP}^{\text{UNIQUE-SAT}}$. We'll use this to get a randomized reduction of PH to $\oplus$P. Because UNIQUE-SAT only cares about formulas with exactly 0 or 1 satisfying assignments, $\oplus$SAT solves UNIQUE-SAT on formulas within the promise. This shows that $\mathrm{NP} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$.

Given this, we use the fact that a problem from the first homework relativizes. Namely, $\mathrm{NP} \subseteq \mathrm{BPP} \Rightarrow \mathrm{PH} \subseteq \mathrm{BPP}$ relativizes. So $\mathrm{NP}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}} \Rightarrow \mathrm{PH}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$. We want to show the hypothesis of this. To get this, we notice that $\mathrm{NP} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$ relativizes as well, giving $\mathrm{NP}^{\oplus \mathrm{P}} \subseteq (\mathrm{BPP}^{\oplus \mathrm{P}})^{\oplus \mathrm{P}}$.

In general, giving an additional oracle O_2 to an oracle machine with access to O_1 can be solved by the base machine by giving it access to oracles O_1 and $O_1^{O_2}$. For the case of $(\mathrm{BPP}^{\oplus \mathrm{P}})^{\oplus \mathrm{P}}$, we get that a BPP machine requires access to a $(\oplus \mathrm{P})^{\oplus \mathrm{P}}$ oracle in addition to a $\oplus$P oracle. These can be combined into a single $(\oplus \mathrm{P})^{\oplus \mathrm{P}}$ oracle using the completeness of $\oplus$SAT, and using the fact that $(\oplus \mathrm{P})^{\oplus \mathrm{P}} = \oplus \mathrm{P}$, we get $\mathrm{NP}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{(\oplus \mathrm{P}^{\oplus \mathrm{P}})} = \mathrm{BPP}^{\oplus \mathrm{P}}$. Having achieved $\mathrm{NP}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$, we conclude that $\mathrm{PH} \subseteq \mathrm{PH}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$.

We point out that the proof given here of this result is simpler than the original proof. This is a demonstration of the power of relativization.

1.1 Deterministic reduction of $\mathrm{BPP}^{\oplus \mathrm{P}}$ to #P

Given a language L in $\mathrm{BPP}^{\oplus \mathrm{P}}$, we wish to determine if $x \in L$ by making a single query to a #P function. We first show that we can separate out the randomized portion of L from the counting portion. This will be useful in performing the reduction.

Definition 1. *Let $\mathcal{C}$ be a complexity class. We define the* BP *operator to give a new complexity class of languages, where* $\mathrm{BP}\cdot\mathcal{C} = \{L | (\exists c > 0)(\exists L' \in \mathcal{C})$

$$\begin{array}{l} x \in L \Rightarrow \Pr_{y\in|x|^c}[\langle x, y\rangle \in L'] > 2/3, \\ x \notin L \Rightarrow \Pr_{y\in|x|^c}[\langle x, y\rangle \in L'] < 1/3 \quad \} \end{array}$$

Notice that $\mathrm{BP}\cdot\mathrm{P} = \mathrm{BPP}$, so this is a reasonable definition of the BP operator. The alternative characterization of $\mathrm{BPP}^{\oplus\mathrm{P}}$ we use is given by the following.

Claim 1. $\mathrm{BPP}^{\oplus\mathrm{P}} = \mathrm{BP}\cdot\oplus\mathrm{P}$.

Proof. We show that for any complexity class $\mathcal{C}$, $\mathrm{BPP}^{\mathcal{C}} = \mathrm{BP}\cdot\mathrm{P}^{\mathcal{C}}$. The result then follows by using $\oplus\mathrm{P}$ as $\mathcal{C}$ and the fact that $\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$.

Consider a BPP machine that can ask queries to a $\mathcal{C}$ language. Without changing the computation, the machine can guess its random bits at the beginning the computation before proceeding. What is left is a $\mathrm{P}^{\mathcal{C}}$ predicate. Now consider a $\mathrm{BP}\cdot\mathrm{P}^{\mathcal{C}}$ language. The $\mathrm{BPP}^{\mathcal{C}}$ machine computing the same language simply generates enough random bits to be the second input of the $\mathrm{P}^{\mathcal{C}}$ machine and then simulates that machine. □

Now consider a language $L \in \mathrm{BP}\cdot\oplus\mathrm{P}$ which we hope to solve in $\mathrm{P}^{\#\mathrm{P}[1]}$. By definition, there is some $f \in \#\mathrm{P}$ such that

$$\begin{array}{l} x \in L \Rightarrow \Pr_{|y|=n^c}[f(x, y) \equiv 1 \pmod 2] > 2/3 \\ x \notin L \Rightarrow \Pr_{|y|=n^c}[f(x, y) \equiv 1 \pmod 2] < 1/3 \end{array}$$

We would like to create a #P function which sums f over all y such that we can detect the gap in probability. The following claim is the main ingredient to make this happen.

Claim 2. *Let $f \in \#\mathrm{P}$. Then there is a function $g \in \#\mathrm{P}$ such that*

$$\begin{array}{l} f(z) \equiv 1 \pmod 2 \Rightarrow g(z, 0^M) \equiv -1 \pmod{2^M} \\ f(z) \equiv 0 \pmod 2 \Rightarrow g(z, 0^M) \equiv 0 \pmod{2^M} \end{array}$$

where $M = 2^m$ is some power of 2.

Before proving this claim, we see how it allows us to complete the construction. Notice that if $f(z) \equiv 1 \pmod 2$, then the last M bits of $-g(z, 0^M)$ are 000...001; if $f(z) \equiv 0 \pmod 2$, the last M bits are 000...000. We would like to sum up $-g(z)$ for all $z = (x, y)$ in such a way that the results do not "spill over" past the last M bits. Because we must sum over 2^{n^c} many y, we need to pick $M > n^c$ to ensure there is no spill over. Namely notice that if $M > n^c$, then

$$\begin{array}{l} x \in L \Rightarrow \sum_{|y|=n^c} -g(x, y, 0^M) \pmod{2^M} > \frac{2}{3}2^{|x|^c} \\ x \notin L \Rightarrow \sum_{|y|=n^c} -g(x, y, 0^M) \pmod{2^M} < \frac{1}{3}2^{|x|^c}. \end{array}$$

In fact, by picking $M > n^c$ we ensure there is a single bit of the sum that we can check to distinguish between the two cases. Thus, $g'(x) = \sum_{|y|=n^c} g(x, y, 0^M)$ is the #P function that we query once to determine if $x \in L$ or not. That is, we compute $g'(x)$, and check if $-g'(x) \pmod{2^M}$ is $> \frac{2}{3}2^{|x|^c}$ or not. Notice that g' is in fact a #P function because a machine can branch on all y and then compute $g(x, y, 0^M)$ on each branch.

All that remains is the proof of the claim.

Proof of Claim 2. The base of the construction is the function $h(z) = 4z^3 + 3z^4$ which has the nice property that

$$z \equiv -1 \pmod{2^M} \Rightarrow h(z) \equiv -1 \pmod{2^{2M}}$$
$$z \equiv 0 \pmod{2^M} \Rightarrow h(z) \equiv 0 \pmod{2^{2M}}$$

Given this property of h, $g(z)$ is the result of applying h recursively $m = \log M$ times to $f(z)$. The claimed property of $g(z)$ follows from the property of h. Consider the running time of the underlying NTM machine whose number of accepting paths is g. Each application of h increases the running time by a constant factor, so overall g is a factor $2^{O(m)} = M^{O(1)}$ slower than f - thus $g \in \#\mathrm{P}$.

All that remains is to prove the claimed property of h. Suppose $z \equiv b \pmod{2^M}$, that is $z = q2^M + b$ for some integer q. Then if we look at the expansion of $h(z)$ and remove all terms with a factor of 2^{2M} or higher power of 2^M, we get $h(z) \equiv 4(3b^2q2^M + b^3) + 3(4b^3q2^M + b^4) \pmod{2^{2M}}$. For $b = 0$, this gives us $0 \pmod{2^{2M}}$, and for $b = -1$, $-1 \pmod{2^{2M}}$. □

2 Interactive Proof Systems

Definition 2 (Interactive Proof Systems)**.** *An interactive proof system for L is a* protocol *(P, V), where P is called the* prover*, and V is called the* verifier*. P is an all-powerful(*i.e.*, no restrictions on its computational resources), randomized Turing machine, and V is a randomized Turing machine running in time polynomial in the input length. In a protocol (P, V), we have:*

1. *P, V have read-only access to the input x.*
2. *P, V have read-only access to separate random-bit tapes.*
3. *P can write messages on tape $T_{P\to V}$ which V has read access to.*
4. *V can write messages to P on tape $T_{V\to P}$ which P has read access to.*
5. *$x \in L$ iff V accepts.*
6. *At any one time, exactly one of V and P is active, and the protocol defines how they take turns being active.*

The protocol (P, V) also has to satisfy the following conditions:

- *completeness(easy to prove membership of members):*

$$x \in L \implies \Pr[(V \leftrightarrow P) \text{ accepts } x] \geq c$$

- *soundness(hard to prove membership of non-members):*

$$x \notin L \implies (\forall P')\Pr[(V \leftrightarrow P') \text{ accepts } x] \leq s$$

where the probabilities are over all coin flips of P and V; $(V \leftrightarrow P)$ refers to an interaction between the P and V; c, s are the completeness *and* soundness *parameters of the interactive proof system, respectively. c is normally set to 2/3 and s to 1/3.*

For the soundness condition, we choose to quantify over all provers since the critical issue is that the verifier itself needs to be sufficiently robust even against "malicious" provers that do not follow the protocol. Note that we can also apply the amplification technique to boost the probability of deciding correctly, and the verifier would still be poly-time. For completeness, $c = 1$ means *perfect completeness* and this means that no true statement is rejected, and is something we strive for.

We make the following observations:

1. Randomness for P is not essential. For each turn it takes, since it is all-powerful, it can figure out the coin flips that will maximize the probability of the verifier accepting, and perform the computation corresponding to those coin flips.

2. On the other hand, the randomness of V is essential, as otherwise, we get only NP. If V is deterministic, then it accepts/rejects with probability 1. Then, using the above fact that we can assume P is deterministic, and that $c > 0$, we get that $(V \leftrightarrow P)$ always accepts members of L. Since V can only read and write a polynomial number of symbols, the length of the transcript of the interaction $(V \leftrightarrow P)$ is polynomial in the length of x, and is a polynomial-length certificate of x's membership. In particular, the sequence of responses of P convinces V of x's membership.

3. If we assume perfect soundness, we get NP as well. This is because $x \in L$ iff there exists a sequence of random "questions" by the verifier and a sequence of "answers" by P that convinces V of x's membership. The converse holds by the assumption of perfect soundness. The other direction has a similar proof as above.

4. Without interaction, since the verifier does not receive any information from the prover, we get BPP.

5. By requiring the verifier's random bit tape to be separate from the prover's, the proofs we defined are actually *private coin* interactive proofs. *Public coin* interactive proofs are where the verifier tosses coins and reveals them to the prover after it's turn is up.

The corresponding complexity class is IP $= \{L | L \textit{ has an interactive proof system}\}$.

3 Power of interaction

Let us now illustrate what we can do with interaction and randomness in the verifier. In particular, we would like to be able to obtain short interactive proofs for problems for which we do not know a short standard proof. One example of such a problem is *graph non-isomorphism*(GNI).

Theorem 1. $GNI \in$ IP

Remember that GI is one of the problems that we think are NP-intermediate. Also, GI has short standard proofs. This theorem by itself will not show that the set of tautologies have short interactive proofs(coNP vs IP). GI is probably not NP-complete, and equivalently, GNI is probably not coNP-complete. In fact, we will give some evidence that GI is not NP-complete.

Before we begin with the proof, we give an intuitive analogy for the way the IP protocol for GNI works. One day, Merlin shows a sword to King Arthur and claims that it is Excalibur. However, Arthur is unconvinced since to him it looks exactly like his own sword. So, Arthur would like

Merlin to convince him that at least he could tell the difference between Excalibur and Arthur's sword. To this end, Arthur turns around, hides the swords and flips a coin. Based on the coin flip, he produces one of the swords and asks Merlin whether it is the purported Excalibur or Arthur's sword. If Merlin answers incorrectly, then Arthur knows that Merlin had lied. Otherwise, he repeats the procedure until he is convinced.

Note that the usage of the name "Arthur" for the verifier and "Merlin" for the prover is standard in the literature, but for public coin systems. The system below uses private coins.

Proof Sketch. Given 2 graphs (G_0, G_1), we would like to show that they are non-isomorphic. An initial idea, based on the analogy above, is to have V flip a coin, produce one of the graphs G based on the coin flip and ask P which of G_0, G_1 it is. But this is too easy since V could compare G with G_0, G_1 by itself. So, instead we have V pick G at random, permute the vertices of G randomly, and ask P to identify G.

Using $\equiv$ to denote isomorphism, if $G_0 \not\equiv G_1$, then P can identify G by trying all permutations on G_0, G_1 and since $G_0 \not\equiv G_1$ exactly one permutation on exactly one of the 2 graphs give G. Otherwise, there exists permutations π, σ such that $\pi(G_0) = \sigma(G_1) = G$, and so P has no way of identifying which of G_0, G_1 was used to produce G. The best it can do then is flip a coin and guess an answer.

So, this protocol satisfies perfect completeness and has $s \geq 1/2$. We can decrease s by increasing the number of rounds. □

We remark that since the verifier is essentially requiring the prover to determine the result of its coin toss, it is crucial that the protocol uses private coins. We next consider interactive protocols that remain secure even when the verifier's random coins are made public. Although the above protocol for GNI does not work in this setting, there is an alternate protocol that does work with public coins.

4 Bounded-round interactive proof systems

We start with the formal definition of AM. In addition to requiring public coins, we limit the number of rounds of interaction between the verifier and prover.

Definition 3. *AM is a public coin system in which Arthur moves first. He flips some coins to determine what to ask Merlin. Arthur then runs a deterministic predicate on the coins, Merlin's reply and the input.*

Formally, a language L is in AM if for some $V \in \mathrm{P}$:

$$x \in L \implies \Pr_{|z|=n^c}[(\exists y \in \Sigma^{n^c}) V(x, y, z)] \geq 2/3$$

$$x \notin L \implies \Pr_{|z|=n^c}[(\exists y \in \Sigma^{n^c}) V(x, y, z)] \leq 1/3$$

Now we present the AM protocol for GNI.

Theorem 2. $GNI \in AM$

Firstly, we review the *Lower Bound Protocol*. We have some subset $S_x \subseteq \{0,1\}^n$ whose size we would like to estimate. We have a universal family of hash functions from n bits to m bits, where the range of the hash functions is roughly the same as the size of S_x. Then, there is a good chance that a hash function chosen randomly from that family will map S_x "evenly" on $\{0,1\}^m$. In particular, a few hash functions from that family will suffice to cover $\{0,1\}^m$ as an image of S_x. The key point is that if S_x is sufficiently big relative to $\{0,1\}^m$, then it is possible, otherwise we need much more functions.

The modifications we will make to the protocol is that we will pick some l^2 points in the range at random, for some parameter l. And now, we would like to determine if we can pick l hash functions such that each of the selected points is in the image of S_x under one of the chosen hash functions. We saw in last lecture that this gives a Σ_2^p predicate.

We can obtain from this an AM protocol to distinguish between the case when $|S_x| \geq 2^m$ and the case when $|S_x| \leq 2^{m-1}$. Arthur picks some points p_i in the range uniformly at random and reveals them to Merlin. Merlin then responds with hash functions h_i and points p_i' in S_x along with certificates of their membership in S_x. Finally, Arthur verifies the membership of points p_i' in S_x and that the image of p_i' under the hash functions indeed covers all the points p_i.

Applying some modifications to the analysis of the original protocol gives us the following results: the AM protocol accepts with probability 1 in the case that $|S_x| \geq 2^m$, and in the other case accepts with probability at most 1/3.

Proof sketch. Define $S_i = \{H | H \equiv G_i\}$ and $S = S_0 \cup S_1$.

The key property that we use is that if $G_0 \equiv G_1$, then $S_0 = S_1$ so $|S| = |S_0| = |S_1|$, and otherwise, $|S| \geq 2\min\{|S_0|, |S_1|\}$.

Since for GNI, we would like to distinguish between 2 sets of size differing by at least a factor of 2, we can adapt the above AM protocol for GNI. However, we need to know the sizes of S_0, S_1, S to use it. By group theory, $|S_i||Aut(G_i)| = n!$ since S_i are the cosets of the subgroup $Aut(G_i)$ in the symmetric group on n points.

Define $T_i = \{(H, \pi) | H \equiv G_i \text{ and } \pi \in Aut(G_i)\}$ and $T = T_0 \cup T_1$. By the above, $|T_i| = n!$. So, if $G_0 \equiv G_1$, $|S| = n!$ and otherwise $|S| = 2n!$.

Since we have a constant factor gap in the sizes, we can use approximate counting to distinguish the 2 sets, such that the larger set corresponds to the non-isomorphic case. We modify it into an AM protocol as above and we are done. □

5 Next Lecture

Next time we'll show that if the graph isomorphism problem is NP-complete then the polynomial hierarchy collapses to the second level. Our proof will use the fact that $GNI \in$ AM. We'll also show that coNP $\subseteq$ IP. The result that we're aiming for is PSPACE = IP.