CS 810: Introduction to Computational Complexity Theory 04/26/2003

Lecture 26: Interactive Proof Systems and Arthur-Merlin Proof Systems

Instructor: Jin-Yi Cai Scribe: Michael DeCoster, Vinod Ganapathy, and Charles Kahn

1 Introduction

In this lecture we see some further applications of the concept of "fat" and "thin" sets. There have been far-reaching applications falling in the class of interactive proof systems.

To start this discussion, let us look at NP. NP can be defined as the class of languages that have short proofs, where the witness can be seen as the proof. Now suppose that we allow probability to enter the picture. Suppose that if $x \in L$, then the algorithm will accept with a high probability, and if $x \notin L$ then the algorithm will not accept with high probability.

The following are the different choices that one can make when allowing probability:

- One-sided ("RP-type") vs. two-sided ("BPP-type") error.
- Who moves first, the prover or the verifier?
- How many rounds? One, two, a bounded (constant, O(1)) number, or a polynomial number of rounds?
- Is the source of randomness shared with the prover (a "public coin") or known only to the verifier (a "private coin")?

In the end, the only thing that probably matters in whether you have a bounded number of rounds or a polynomial number of rounds. This is because anything that has a bounded number of rounds can be collapsed to either 1 or 2 rounds, and because BPP vs. RP doesn't matter as will be shown later in this lecture.

Definition 1 (Verifier). *The verifier is a probabilistic polynomial time Turing machine.*

Definition 2 (Prover). *The prover is not bounded by any computational constraints (he is "all powerful").*

2 The Graph Non-Isomorphism Problem

The Graph Non-Isomorphism (GNI) problem is defined as follows: Given graphs G and H, show that G is not isomorphic to H. It is not clear that there is a polynomially long proof for this problem, but this does not necessarily mean that $GNI \notin NP$. We are also not claiming that, for any two specific pairs of graphs GNI is hard.

We don't know of any polynomial time predicate $P(\cdot,\cdot,\cdot)$ s.t. $G \ncong H \Leftrightarrow (\exists^p z)[P(G,H,z)]$.
Now the question is, How do I (as a prover) convince you (as a verifier) that $G \ncong H$ is indeed the case?

The verifier can randomly pick either G or H, randomly choose a permutation of that choice, and then give the prover the new graph. Now the prover needs to tell the verifier which graph was originally picked. Consider the following argument to show that this is indeed useful.

Assume G and H are actually isomorphic. Then there is a permutation σ s.t. $G = \sigma(H)$. Then $\pi(G) = \pi(\sigma(H))$, where π runs through all $n!$ permutations of the graph it was given. Thus every permutation of G or H has equally likely probability of being presented, regardless of which was originally chosen. Therefore, the probability of the prover getting it right if $G \cong H$ is $\frac{1}{2}$.

However, if G and H are not isomorphic, then the new graph will be isomorphic to either G or H, but not both. Thus the prover will always choose the correct graph if the prover isn't lying. If this is run 1000 times, and the prover is right each time, the probability of $G \cong H$ is only 2^{-1000}, so the verifier can be very confident the prover is telling the truth.

The exact protocol for this is as follows:
<u>Protocol</u>
Verifier: Secretly pick k choices of either H or G. Apply a random permutation to each of them, and present to prover.
Prover: Produce a sequence of k bits to indicate the choices made in picking G and H by the verifier.
Verifier: Accept iff all choices are correct.

Now we will present this protocol stated more formally:
Given G_0, G_1:
V: Pick $s \in_R \{0,1\}^k$
$\forall 1 \le i \le k$, independently and randomly pick a permutation π_i.
Send to P: $< \pi_1(G_{s_1}), \pi_2(G_{s_2}), \cdots, \pi_k(G_{s_k}) >$.
P: Reply with $s' \in \{0,1\}^k$
V: Accept if and only if: $s = s'$

The algorithm for the prover will be as follows:
If G_0, G_1 are such that $G_0 \ncong G_1$, then $\exists$ a prover P_1, s.t. Upon receiving $< \pi_1(G_{s_1}), \pi_2(G_{s_2}), \cdots, \pi_k(G_{s_k}) >$, runs an isomorphism test $\pi_i(G_{s_i}) \cong G_0$ or G_1. Exactly one of the two will be isomorphic, so P_1 replies with $s' = (s'_1, s'_2, \cdots, s'_k)$ s.t. $G_{s'_i} \cong \pi_i(G_{s_i})$
Pr[V accepts with P_1]=1 if $G_1 \ncong G_2$
If $G_0 \cong G_1$ the verifier does not know if he interacts with a honest prover or not, so we will quantify over all provers.
If $G_0 \cong G_1$ $\forall P^*$ Pr[V accepts with P^*]$\le \frac{1}{2^k}$.

This has shown the idea of a language with an interactive proof. The choices that were made in this interactive proof were as follows:

1. verifier has a private coin
2. one sided error
3. one-round protocol: merged all the questions using k
4. verifier moves first

From this example, it seems that private coins were necessary, but we will find that things will work just as well using public coins.

Definition 3 (Interactive Proof System). *An interactive proof system is a protocol, an interacting pair of funtions, specifying up to a polynomial number of rounds of interaction. L is said to have in interactive proof system if $\exists(P, V)$ where V is computable by a probabilistic polynomial time TM, and P is any function.*

$$x \in L \Rightarrow \Pr[Acc(P, V, x)] = 1,$$
$$x \notin L \Rightarrow \forall P^* \Pr[Acc(P^*, V, x)] < \epsilon,$$

for fixed constant $\epsilon \in (0, 1)$.

3 Arthur Merlin Games

In the previous section, we saw how the prover could convince the verifier with high probability that two graphs G_1 and G_2 were not isomorphic. However, the success of that protocol, as we defined it, depended crucially on the fact that the verifier had access to a secret random source, and that he could generate a set of random coin tosses which would be used to select the graph to be permuted and sent over to the prover. In fact, if the random source was not secret to the verifier, and the prover could see the series of coin tosses made by the verifier, the prover could always win. The prover just has to send back the random bit chosen by the verifier, and this answer will always correspond to the graph chosen by the verifier.

In this section, we will consider a variation of the above protocol, where the verifier's random source is not secret to the verifier and the prover can also see the series of coin tosses as made by the prover. Following convention, we will call the prover *Merlin*, mythically an all powerful magician, and call the verifier *Arthur*. As usual, Arthur is supposed to be a probabilistic polynomial time Turing machine, and Merlin is assumed to be an all powerful Turing machine, i.e., we place no limits on his computational resources. In fact, in this setting Merlin is even more powerful in some sense, since he has access – or is powerful enough to simulate – the random source used by Arthur. Since Arthur cannot hide his random source from Merlin anyway, we will assume for the rest of our discussion that Arthur always sends the random numbers that he used in his computation to Merlin. Furthermore, we will assume that Merlin can always simulate Arthur's computation to compute the next query based on the history of computation, and the random numbers received by Arthur. Thus, in effect, Arthur does not have to send anything to Merlin except the random numbers generated by him. Arthur's only task is to verify whether the computation satisfies a predetermined condition, and whether to accept the input.

Definition 4 (Class AM). *We say that $L \in \mathtt{AM}$ if and only if there is a deterministic polynomial time computable predicate P such that:*

$$x \in L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}}[\exists^p y P(x, r, y)] = 1$$
$$x \notin L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}}[\exists^p y P(x, r, y)] \leq \epsilon$$

Notice that in the definition of $\mathtt{AM}$, we have an $\mathtt{NP}$-like question, and we consider the probability (over the random string r) whether the question is satisfied or not. Suppose instead we had a deterministic polynomial time computable predicate Q, as follows:

$$x \in L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [Q(x,r)] = 1$$
$$x \notin L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [Q(x,r)] \leq \epsilon$$

This class is in fact $\mathtt{coRP}$. As a result, the class $\mathtt{AM}$ is also often referred to as $\mathtt{coR{\cdot}NP}$.

Now that we have defined $\mathtt{coR{\cdot}NP}$, we can ask what the class $\mathtt{R{\cdot}NP}$ is:

Definition 5 (Class $\mathtt{R{\cdot}NP}$). *We say that $L \in \mathtt{R{\cdot}NP}$ if and only if there is a deterministic polynomial time computable predicate P such that:*

$$x \in L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] \geq 1 - \epsilon$$
$$x \notin L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] = 0$$

We claim that this class is in fact $\mathtt{NP}$.

Lemma 1. $\mathtt{R{\cdot}NP} = \mathtt{NP}$

Proof. $\mathtt{R{\cdot}NP} \subseteq \mathtt{NP}$: Rewrite $\exists^p y P(x,r,y)$ as $\exists^p (r,y) Q(x,(r,y))$. Then

$$\Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] \geq 1 - \epsilon \Longleftrightarrow \exists^p (r,y) Q(x,(r,y))$$
$$\Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] = 0 \Longleftrightarrow \nexists^p (r,y) Q(x,(r,y))$$

Hence if for a language L, $L \in \mathtt{R{\cdot}NP}$, then $L \in \mathtt{NP}$.

$\mathtt{NP} \subseteq \mathtt{R{\cdot}NP}$: Consider a language $L \in \mathtt{NP}$. This means that if $x \in L$, then $\exists^p y Q(x,y)$, where Q is a polynomial time deterministic predicate. This is the same as saying $\exists^p y P(x,r,y)$, where $P(x,r,y) = Q(x,y)$. Hence, $\Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] = 1 \geq 1 - \epsilon$. Similarly, if $x \notin L$, then $\nexists^p y Q(x,y)$. This is the same as saying $\nexists^p y P(x,r,y)$. So $\Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] = 0$.

Hence, $\mathtt{NP = R{\cdot}NP}$. ∎

We can also define the class $\mathtt{BP{\cdot}NP}$.

Definition 6 (Class $\mathtt{BP{\cdot}NP}$, also known as $\mathtt{AM}_2$). *A language $L \in \mathtt{BP{\cdot}NP}$ if and only if:*

$$x \in L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] \geq 1 - \epsilon$$
$$x \notin L \Rightarrow \Pr_{r \in \{0,1\}^{p(|x|)}} [\exists^p y P(x,r,y)] \leq \epsilon$$

This class is also often referred to as $\mathtt{AM}_2$, the version of $\mathtt{AM}$ with two-sided error. We'll show that this class and $\mathtt{AM}$ are in fact the same.

Lemma 2. $\mathtt{AM} = \mathtt{AM}_2$

Proof. $\texttt{AM} \subseteq \texttt{AM}_2$: This part is obvious, since $\Pr_{r\in\{0,1\}^{p(|x|)}}[\exists^p y P(x,r,y)] = 1$ implies $\Pr_{r\in\{0,1\}^{p(|x|)}}[\exists^p y P(x,r,y)] \geq 1-\epsilon$.

$\texttt{AM}_2 \subseteq \texttt{AM}$: Consider a language $L \in \texttt{AM}_2$. From the definition of $\texttt{AM}_2$, we have,

$$x \in L \Rightarrow \Pr_{r\in\{0,1\}^m}[\exists^p y P(x,r,y)] \geq 1 - \frac{1}{p(m)}$$
$$x \notin L \Rightarrow \Pr_{r\in\{0,1\}^m}[\exists^p y P(x,r,y)] \leq \frac{1}{p(m)}$$

where $p(m)$ is some polynomial in m, and $m = poly(|x|)$. We obtained the above definition by taking $\epsilon = \frac{1}{p(m)}$. We note that this can always be done by boosting confidence by repeated trials. The above definition can be interpreted as saying that the set of witnesses r that satisfy $\exists^p y P(x,r,y)$ is *fat*. At this point, we recall two lemmas that we proved last class.

Lemma 3. *If a set $S \subseteq \{0,1\}^n$ is $\frac{1}{m}$-fat, then if $S \oplus u_i$ denotes the set of vectors S shifted by the vector u_i, we have:*

$$\Pr_{u_1,u_2,\dots,u_m}\left[\bigcap_{i=1}^{m} S \oplus u_i \neq \emptyset\right] = 1$$

If on the other hand S is thin, then we have:

$$\Pr_{u_1,u_2,\dots,u_m}\left[\bigcap_{i=1}^{m} S \oplus u_i \neq \emptyset\right] \leq \frac{2^n}{m^m}$$

Based on this lemma, we will prove the thoerem. Fix an input x. Let $S = \{r | \exists^p y P(x,r,y)\}$. Clearly, if $x \in L$ for $L \in \texttt{AM}_2$, then the set S is fat. However, if $x \notin L$, then the set S is thin.

We shall construct a new one-sided protocol as follows: Arthur will give a set of random numbers to Merlin. To convince Arthur that $x \in L$, Merlin will try to prove that the set S is fat. Arthur will send a set of strings $r_1, r_2, \dots r_m$ to Merlin. If S is fat, then the sets $S \oplus r_i$ have a non-empty intersection. This is equivalent to saying that if we consider one intersection point, say r, then $r \oplus r_i \in S$, which is equivalent to saying that $r \oplus r_i$ must be such that $\exists y P(x, r \oplus r_i, y)$. So Merlin sends back the r and $y_1, y_2, \dots y_m$ such that each $P(x, r\oplus r_i, y_i)$ is satisfied. If on the other hand $x \notin L$, the probability that r exists is extremely small, in fact $\leq \frac{2^n}{m^m}$, which we call ϵ.
Hence,

$$x \in L \Rightarrow \Pr_{r_1,r_2,\dots,r_m}\left[\bigwedge_i P(x, r\oplus r_i, y_i)\right] = 1$$
$$x \notin L \Rightarrow \Pr_{r_1,r_2,\dots,r_m}\left[\bigwedge_i P(x, r\oplus r_i, y_i)\right] \leq \epsilon$$

which gives us the desired one-sided error protocol, and hence shows that $L \in \texttt{AM}$.

□