CS 710: Complexity Theory 12/09/2011

Lecture 26: Arthur-Merlin Games

Instructor: Dieter van Melkebeek Scribe: Chetan Rao and Aaron Gorenstein

Last time we compared counting versus alternation and showed that #P could be approximated by Σ_2, not just Σ_3. We also demonstrated the power of relativization by adapating a result from HW1. This lead us to conclude that $\mathrm{NP}^{\oplus \mathrm{P}} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$.

Today we will conclude our discussion on the power of counting, and complete the proof by demonstrating the inclusions $\mathrm{PH} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$ and $\mathrm{BPP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$.

We will also 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.

We will conclude with the power of *bounded*-rounds interactive proof systems, formalized as Arthur-Merlin games.

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

Recall the overall view of our result:

1. $\mathrm{PH} \subseteq \mathrm{BPP}^{\oplus \mathrm{P}}$
2. $\mathrm{BPP}^{\oplus \mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}}$

We showed 1 last time, and today we will show 2. In doing so we will exploit the closure properties of #P. First, however, we will isolate and consider the BP operator on complexity classes.

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

$$\begin{aligned} x \in L &\Rightarrow \Pr_{|y|=n^c}[f(x,y) \equiv 1 \pmod 2] > 2/3 \\ x \notin L &\Rightarrow \Pr_{|y|=n^c}[f(x,y) \equiv 0 \pmod 2] < 1/3 \end{aligned}$$

We would like to create a $\#\mathrm{P}$ function which sums f over all y such that we can detect the gap in probability. Our key claim is the following:

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

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

where m is itself some power of 2 and $z = \langle x, y \rangle$ from our definition of BP*.*

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 $z = \langle x, y \rangle$ 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

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

In other words, if $m > n^c$ and $h(x) = \sum_{|y|=|x|^c} g(z, m)$ then

$$\begin{aligned} x \in L &\Rightarrow -h(x) \mod 2^m \in [\tfrac{2}{3}2^{n^c}, 2^{n^c}] \\ x \notin L &\Rightarrow -h(x) \mod 2^m \in [0, \tfrac{1}{3}2^{n^c}] \end{aligned}$$

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. We claim that the sum, and $h(x)$, are in $\#\mathrm{P}$. It is just a uniform sum, and $\#\mathrm{P}$ is closed under that.

All that remains is the proof of the key claim. We will be doing something slightly different: we will claim that $\exists g \in \mathrm{GapP}$, and $h(x) \in \mathrm{GapP}$. This seems odd: wouldn't that mean there are 2, rather than 1, query to the $\#\mathrm{P}$ oracle? We will show how to resolve that.

Proof of Claim 2. Say $h = h_+ - h_-$, we break up h into its two components. Clearly that function is in GapP. We re-write $h_-(z) = \#\mathrm{Acc}_M(z)$—the number of accepting paths. Note how we can trivially fix the length of a TM's computation path. So we know that the full number of computation

paths is $\#\text{Acc}_M(z) = 2^{|z|^d} - \#\text{Rej}_M(z)$. Clearly that negative value is a #P query. Now we can redefine $h(x)$ as a sum of two #P queries, meaning it is in #P:

$$h(x) = \#\text{Acc}_{M_+}(z) + (\#\text{Rej}_{M_-}(z) - 2^{|z|^d})$$

All that remains now, is to find some f that fits the initial requirements we describe when we introduced our key claim, so that we can build g from it.

We will show that, given $k \mod 2^m$, we can have a transformation which preserves:

$$t \equiv -1 \mod 2^m \Rightarrow Q(t) \equiv -1 \mod 2^{2m}$$
$$t \equiv 0 \mod 2^m \Rightarrow Q(t) \equiv 0 \mod 2^{2m}$$

We want Q to be a constant-degree polynomial, $Q \in \mathbb{Z}[t]$.

Once we have Q, we're done, we can simply apply it repeatedly. If $m = 1$, then we only need a logarithmic number of doubles from applying Q. Because $Q \in \mathbb{Z}(t)$, if $t \in$ GapP we're still in GapP. The running time is multiplied by $\deg(Q)$ for each application of the operation, and we only apply the operation a logarithmic number of times. That is only a polynomial overhead factor, so this works. Basically, we will begin with $t = f(z)$, and then apply Q $\log(m)$ times to get g. Does there exist such a Q?

Say $t \equiv b + q2^m \mod 2^m$ (with $b = -1$ or 0). Consider simply squaring the equation: $t^2 \equiv b^2 + 2bq2^m \mod 2^{2m}$, and other terms are modded out. In this first case, if $t = 0$, clearly we are still $\equiv 0$ under this new mod. But that is not the case for $t = -1$. Now consider $t^3 \equiv b^3 + 3b^2q2^m$, and the other terms are modded out. If you consider

$$Q(t) = -2t^3 - 3t^2 \tag{1}$$

Then you will find that both 0 and -1 are preserved! Hence, that's our new polynomial. Note that we needed GapP for the negative coefficients. That concludes the proof. □

By allowing Q to go up to degree 4, we can get a different polynomial that also works that also only has nonnegative coefficients. Such a polynomial obviates the need to use GapP, requiring only #P.

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 $= \{L | L \text{ has an interactive proof system}\}$.

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

The key idea for the transformation from private-coin protocols to public-coin protocols can be found in the *Lower Bounds Protocol*: It is a public-coin protocol for the set lower bound problem,

which is a promise problem with positive instances I_p and negative instances I_n:

$$I_p = \left\{ \quad \langle x, a \rangle \quad \middle| \quad |S_x| \geq a \quad \right\}$$
$$I_n = \left\{ \quad \langle x, a \rangle \quad \middle| \quad |S_x| \leq (1-\epsilon)a \quad \right\}$$

Here, $\epsilon = \epsilon(n) = \frac{1}{\text{poly}(n)}$.

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.

Recall that we made the Π_2 predicate, $\forall z \exists h_{\ldots}$. We observed that in the "yes" case, it was true for all z, so regardless of the hash functions chosen it would work. In the "no" case, *most* of the z don't hold.

That precisely gives us the IP protocol we want. The public coinflips are analagous to the z in our approximate counting setting.

This gives us the AM protocol, where the coins are public. Now we would like to demonstrate that $GNI \in$ AM.

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

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. □

In the case where Arthur makes some choices, and Merlin responds, and then Arthur concludes with a deterministic decision procedure is AM. Other problems can be formulated as Merlin going first, that is MA. We can denote the "rounds" as AMA... and consider if more rounds adds more power.

5 Next Lecture

Next lecture we will show that number of rounds does not matter, that MA$\subseteq$AM, and so with some additional reasoning conclude that all bounded interaction IP is in AM.

We will also demonstrate that IP=PSPACE, and introduce the PCP theorem, a powerful variant of this idea.

6 Acknowledgements

Many thanks to Jeff Kinne and Ashutosh Kumar for their notes from Spring 2010.