CS 810: Complexity Theory	3/26/2007

Lecture 23: Interactive Proofs

Instructor: Dieter van Melkebeek — Scribe: Seeun William Umboh

Last time, we introduced the notion of proof complexity, where we looked at proof systems for the set of propositional tautologies. We required the proofs to be efficiently verifiable so proofs correspond to non-deterministic computations. So, a language L has short proofs of membership if and only if L is in NP. In particular, if propositional tautologies have short proofs, then NP = coNP.

Today, we relax the notion of proof by allowing randomness in verification, and allow more interaction between the *prover* and *verifier*. In contrast, previously, the verifier was deterministic; the interaction was limited to a single round and was one-way: the prover passes the purported proof to the verifier and the verifier accepts or rejects based on that proof. With the new notion of proof, L has short proofs if and only if L is in PSPACE, and so tautologies have short proofs, unlike in the previous definition of proof.

1 Interactive Proof Systems

Definition 1 (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 $\mathrm{IP} = \{L | L \textit{ has an interactive proof system}\}$.

2 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. $GNI \in \mathrm{IP}$

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. $\square$

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.

3 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 2. *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. From lecture 20, 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 $T_i = \{(H, \pi) | H \equiv G_i \text{ } and \text{ } \pi \in Aut(G_i)\}$ 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. □

Note that the above protocol has perfect completeness. So, one question we may ask is whether the conditions of public coins and perfect completeness place any restrictions on the power of interactive proof systems. The answer is negative and in fact:

Theorem 3. *For any interactive proof system with $r(n)$ rounds, there exists an equivalent interactive proof system with:*

1. *Perfect completeness using $r(n)$ rounds.*

2. *Perfect completeness and public coins using $r(n) + 2$ rounds.*

We omit the proof but the idea for (1) is that we can use the hashing technique to eliminate error on the positive side.

Now, we focus on bounded-round public coin systems.

Definition 3. *MA is a public coin system in which Merlin moves first, and then Arthur flips some coins and runs a deterministic predicate on the coins, Merlin's message and the input.*

Formally, a language L is in MA if:

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

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

We can extend AM and MA in a similar way that Σ_2^p extends NP: AMA, MAM, ... This gives a hierarchy of classes based on the number of rounds allowed, and whether Arthur or Merlin goes first.

However, unlike the polynomial hierarchy, this hierarchy collapses. It may seem surprising since Merlin behaves like an existential quantifier, and Arthur behaves like a universal quantifier with restricted power. We now prove the key ingredient for the collapse:

Theorem 4. MA $\subseteq$ AM

One of the main ideas behind the proof is that MA and AM are somewhat similar to a randomized NP machine. For example, in MA, the prover provides a proof and the verifier verifies it probabilistically instead of deterministically like in NP.

Proof. For $L \in$ MA, we have:

$$x \in L \implies (\exists y \in \Sigma^{n^c}) \Pr_{|z|=n^c}[V(x, y, z) = 1] \geq 2/3$$

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

where V is the underlying deterministic predicate that the verifier evaluates.

So, to get MA $\subseteq$ AM, we would like to swap the choice of the purported proof and the choice of the random string. For the case where $x \in L$, an examination of the condition on the RHS reveals this swap is not a problem. But for the case when $x \notin L$, the swap of quantifiers does not work out. We solve this problem by using amplification and the union bound to our advantage. In particular, for a fixed y, we amplify V such that when $x \notin L$, $Pr_z[V(x, y, z)] \leq \frac{1}{3} \cdot 2^{-n^c}$. Given this, we can now swap the quantifiers, to get

$$x \in L \implies \Pr_{|z|=n^d}[(\exists y \in \Sigma^{n^c}) V(x, y, z) = 1] \geq 2/3$$

$$x \notin L \implies \Pr_{|z|=n^d}[(\exists y \in \Sigma^{n^c}) V(x, y, z) = 1] \leq \frac{1}{3} \cdot 2^{-n^c} \cdot 2^{n^c} = \frac{1}{3}$$

where n^d is some polynomial of n^c.

This gives us an AM protocol and we are done. ∎

Equipped with this theorem, intuitively we can switch the order of the first 2 turns in MAM to get AMM and then merge the 2 Merlin turns into 1, and thus MAM = AMM = AM. Similarly, AMA = AAM = AM. We omit the formal proof.

In fact, we can prove something stronger for bounded-round protocols. By the above process, we can halve the number of rounds needed:

Theorem 5. *For any poly-time $r(n)$ with $r(n) \geq 2$,* $\mathrm{AM}(2r(n)) = \mathrm{AM}(r(n))$.

Proof. Omitted. $\square$

Corollary 1. $\mathrm{AM}(k) = \mathrm{AM}$ *for all constant k.*

Thus, this corollary and Theorem 3 give us that private coin systems with a constant number of rounds are equivalent to AM. Note that this does not imply that IP collapses to AM because we would need a super-constant number of applications of the above. Intuitively, this is due to a polynomial factor blow-up in the verification procedure when we apply the above. So, when we apply it only a constant number of times, the verification still runs in polynomial time, but not if we require super-constant number of applications.

3.1 Relationship with other complexity classes

We have the following facts:

1. $\mathrm{MA} \subseteq \Sigma_2^p$
2. $\mathrm{AM} \subseteq \Pi_2^p$
3. $\mathrm{AM} \subseteq \mathrm{NP}/\mathrm{poly}$
4. Under reasonable derandomization hypotheses, $\mathrm{AM} = \mathrm{NP}$.

(1) follows from the fact that Arthur's computation is a BPP computation on Merlin's proof and the input, and we can simulate BPP using approximate counting with a Σ_2^p predicate P on (x, y). Then, since we can simulate Merlin's computation with an existential quantifier, we get $(\exists y \in \Sigma^{n^c})[P(x, y) = 1]$. Merging the existential quantifier with the one in P, we get a Σ_2^p predicate. (2) follows similarly, but we use the fact that $\mathrm{BPP} \subseteq \Pi_2^p$ instead.

(3) follows using a proof similar to the one for $\mathrm{BPP} \subseteq \mathrm{P}/\mathrm{poly}$. We use amplification to obtain a random string that will work for all the inputs of a fixed length, and then use that string as an advice. We leave the details as an exercise.

The intuition behind (4) is that AM is somewhat similar to NP. The proof is by relativizing the proof that E has $2^{\Omega(n)}$-size circuits implies that P = BPP. (4) also suggests that GI is not NP-complete, since otherwise GNI would be coNP-complete and since $GNI \in \mathrm{AM} = \mathrm{NP}$ under reasonable hypotheses, that would imply that NP = coNP. In fact, we get a collapse of PH if GI is NP-complete even without a derandomization assumption.

Theorem 6. *GI is not NP-complete unless* $\Sigma_2^p = \Pi_2^p$.

Proof. Suppose that GI is NP-complete. Then, GNI is coNP-complete.

Let L be a Σ_2^p language. Then, we have that

$$x \in L \iff (\exists y \in \Sigma^{n^c})(\forall z \in \Sigma^{n^c})[< x, y, z > \in V]$$

for some constant c and a polynomial-time predicate V. Since GNI is coNP-complete and $GNI \in$ AM, we can reduce the $(\forall z \in \Sigma^{n^c})[< x, y, z > \in V]$ coNP predicate as an AM protocol. Since we can express the existential quantifier as a Merlin phase, we have that $\Sigma_2^p \subseteq \mathrm{MAM}$. But because all constant-round interactive proof systems collapse to AM, $\Sigma_2^p \subseteq \mathrm{AM} \subseteq \Pi_2^p$. Thus, $\Sigma_2^p = \Pi_2^p$. $\square$

4 coNP is in IP

One of the main reasons we have for studying the class IP is to see if we can obtain short proofs using interactive proof systems in settings where short standard proofs are unlikely, such as tautologies. While it is unlikely that coNP $\subseteq$ AM, since the above proof would then show that $\Sigma_2^p = \Pi_2^p$, we do have that coNP $\subseteq$ IP. Note that this is one of the few non-trivial results in complexity theory that does not relativize.

Theorem 7. *The language* $L = \{(\varphi, \#SAT(\varphi)) | \varphi \in \{0,1\}^*\}$ *is in* IP*, where* $\#SAT(\varphi)$ *denotes the number of assignments satisfying a* SAT *clause* φ*.*

Note that L is the decision variant of the counting problem for SAT.

Proof(first half). For L, we are given (φ, k) and we would like to verify using an interactive proof system that $\#SAT(\varphi) = k$. One of the key ingredients is arithmetization, which we looked at briefly in the last lecture.

Assume that φ has n variables and m clauses, and is 3-CNF. So, we would like to transform $\varphi(x_1, \ldots, x_n)$ into a polynomial over the integers, $\tilde{\varphi}(x_1, \ldots, x_n)$ such that $\tilde{\varphi}$ agrees with φ on $\{0,1\}^n$ and $\tilde{\varphi}$ has degree at most m in every variable and easy to evaluate as well.

Now we show how to transform a clause C. For example, if $C = (x_1 \vee \bar{x_2} \vee x_3)$, then after transformation we have $\tilde{C} = 1 - (1 - x_1)x_2(1 - x_3)$. Then, to construct $\tilde{\varphi}$, we use $\tilde{\varphi} = \prod_{j=1}^{m} \tilde{C}_j$. Since each variable occurs at most once in each clause, the construction gives a polynomial with degree at most m in every variable, and also $\tilde{C}_j = 1$ if C_j is satisfied and $\varphi = 1$ if and only if every $\tilde{C}_j = 1$. Thus, $\tilde{\varphi}$ agrees with φ on $\{0,1\}^n$. Also, $\tilde{\varphi}$ is easy to evaluate since we just have to go through all the transformed clauses $\tilde{C}_j$ and make sure they all evaluate to 1.

Then, to count the number of satisfying assignments, we use $\sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} \tilde{\varphi}(x_1, \ldots, x_n)$. □

5 Next Lecture

For next time, we will finish the above proof. Then, we will present a *sumcheck protocol* with perfect completeness to verify that degree d polynomials have d zeros and then show that IP = PSPACE. We will also introduce PCP, the class of languages with *probabilistically checkable proofs* and the power of being able to "interrogate" multiple provers with different "questions", with each prover able to access only its own histories. This is akin to police interrogation in which there are multiple suspects, each of which is isolated from the others, and the police can ask different questions to each.