CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 17

March 31, 2010 **Scribe: Jonathan Ullman**

1 Interactive Proofs

Recall the definition of **NP**: $L \in \mathbf{NP} \iff$ there exists a polynomial-time V and polynomial p s.t.

Completeness: $x \in L \implies \exists \pi \in \{0,1\}^{p(|x|)} V(x,\pi) = 1$,

Soundness: $x \notin L \implies \forall \pi \in \{0,1\}^{p(|x|)} V(x,\pi) = 0$.

This definition is essentially the definition logicians have always used for proof systems, except that complexity theorists have formalized the requirement that the proof be easy to check.[1]

Interactive proofs add two new ingredients to classical proof systems:

Randomness: The verifier can toss coins and is allowed to err with some (small) probability if it is unlucky in its coin tosses.

Interaction: Instead of a static proof π, the efficient verifier can *interact* with a computationally unbounded prover.

To formalize this, we must first say what it means for two algorithms (the verifier and prover) to interact. Each algorithm is interpreted as a *next-message function* mapping the party's input x, its coin tosses r, and the prior messages $m_1, \ldots, m_{i-1}$ to the party's next message m_i or `halt/accept/reject`. We write $\langle P, V \rangle (x)$ to indicate the probability space where we choose both parties' coin tosses uniformly at random, and alternately apply their next-message function until one party halts.

Definition 1 (Interactive Proof) *An* interactive proof *for a language L is a pair of interactive algorithms $\langle P, V \rangle$ algorithms s.t. $\forall x$*

Efficiency: *The total length of communication in $\langle P, V \rangle (x)$ is at most* $\mathrm{poly}(|x|)$ *and V runs in time* $\mathrm{poly}(|x|)$*,*

Completeness: $x \in L \implies \Pr[V \textit{ accepts in } \langle P, V \rangle (x)] \geq 2/3$*,*

Soundness: $x \notin L \implies \forall P^* \ \Pr[V \textit{ accepts in } \langle P^*, V \rangle (x)] \leq 1/3$*,*

where all the probabilities are taken only over the coin tosses of V.

[1] The definition of **NP** also bounds the length of the proof to be polynomial in the size of the statement, but this can be addressed by padding. For example, the language consisting of pairs $(\varphi, 1^n)$ such that φ is a mathematical statement (e.g. in set theory) with a proof of length at most n (e.g. in Zermelo-Frankel set theory) is **NP**-complete. If we don't impose *any* bound on the length of the proof, we run into undecidability.

Note that we require that soundness hold *no matter what strategy P^* the prover follows.* This captures the idea that we do not have to trust the prover to be convinced by the proof.

Note also that since the probabilities are only over the coin tosses of V, we can reduce the error through repetition. This is relatively easy to show for sequential repetition, but also holds for parallel repetition (which has the advantage of not increasing the number of rounds of interaction).

Definition 2 (IP)

$$\mathbf{IP} = \{L \mid L \textit{ has an interactive proof.}\}$$

Clearly $\mathbf{NP} \subseteq \mathbf{IP}$ where the prover simply sends the **NP** witness to the verifier. The rest of this lecture will show that **IP** is actually much larger:

2 Interactive Proof of Graph Non-isomorphism

The first evidence we will provide that **IP** is larger than **NP** is an interactive proof of graph non-isomorphism, which is not known to be in **NP**.

2.1 Graph Non-isomorphism

Let $G = ([n], E)$ be a graph and $\pi \in S_n$ be a permutation of $[n]$. Let

$$\pi(G) = ([n], \{(\pi(i), \pi(j)) \mid (i, j) \in E\})$$

be a new graph with the labels of the vertices permuted by π. We say that two graphs G and H are *isomorphic*, written $G \cong H$, if there exists $\pi \in S_n$ s.t. $\pi(G) = \pi(H)$. We define the language

$$\text{GNI} = \{(G_0, G_1) \mid G_0 \ncong G_1\}.$$

The language $\text{GI} = \overline{\text{GNI}}$ is in **NP** but GNI is not known to be in **NP**. However, we will see that GNI does have an efficient interactive proof system.

2.2 GNI $\in$ **IP**

Prover	(G_0, G_1)	Verifier
		$b \overset{R}{\leftarrow} \{0,1\}$ $\pi \overset{R}{\leftarrow} S_n$ $H := \pi(G_b)$
	$\longleftarrow H$	

if $H \cong G_0$ **then**
$c := 0$
else
$c := 1$
end if
	$c \longrightarrow$	

		if $c = b$ **then**
		accept
		else
		reject
		end if

We will verify that this proof satisfies efficiency, completeness, and soundness:

Efficiency: The verifier clearly runs in time polynomial in the size of the input.

Completeness: if $G_0 \ncong G_1$ then H is isomorphic only to G_b, thus the correct prover will always return $c = b$ and V accepts with probability 1.

Soundness: If $G_0 \cong G_1$, then the distribution of $\pi(G_0)$ and $\pi(G_1)$ are identical. Thus for every P^*, $\Pr[P^*(G_0, G_1, H) = b] \leq 1/2$. If we want to be exact about the $1/3$ in the soundness requirement, we could run this protocol twice.

Some remarks on the interactive proof system for GNI:

- In this protocol, it is essential that V have "private coins," meaning that V can hide the choice of b from the prover. However, it will turn out that every language with an interactive proof has one where the verifier uses only "public coins."
- The protocol is very efficient, it uses only two rounds of communication.
- The prover only needs to be able to decide GNI to run this protocol.

3 Motivation for Interactive Proofs

- Interactive proofs are a natural model of proofs in every day interactions, like a courtroom.

- Interactive proofs can have additional properties that are not possible for standard **NP** proofs, such as being "zero knowledge" (where the verifier learns nothing other than the fact that the assertion being proven is true).
- Many situations in cryptographic protocols can be modeled as interactive proofs. For these kinds of applications, we need a prover that is polynomial time when given an **NP** witness, and thus the benefit of interactive proofs is not handl ing languages outside of **NP** but rather the additional properties they enable (like zero knowledge).
- Secure outsourcing of computation. For example, we may have a prover than runs in cubic time and a verifier that only runs in linear time.
- Variants of interactive proofs (multiprover interactive proofs, probabilistically checkable proofs) are closely related to hardness of approximation for **NP** optimization problems (as we will see in a few weeks).

4 IP Contains Counting

In this section we will prove that **IP** contains $\mathbf{P}^{\#\mathbf{P}}$. This is a surprising result, as it's not even obvious that $\mathbf{coNP} \subseteq \mathbf{IP}$.

Theorem 3 $\mathbf{P}^{\#\mathbf{P}} \subseteq \mathbf{IP}$. *In particular,* **IP** *contains all of* **PH**.

First we will outline one attempt at the proof, to give intuition for the correct interactive proof. Note that it is sufficient to give an interactive proof system for the following (**PP**-complete) decisional version of #SAT.

$$\#\mathrm{SAT}_D = \left\{ (\varphi, k) \text{ s.t. } \sum_{b_1,\ldots,b_n \in \{0,1\}^n} \varphi(b_1, \ldots, b_n) \geq k \right\}.$$

The first thing we might try is as follows:

(φ, k)

Prover		**Verifier**

$N = \sum_{b_1,\ldots,b_n \in \{0,1\}} \varphi(b_1, \ldots, b_n)$	$N \longrightarrow$	**if** $N < k$ **then**
		  reject
		end if

$N_0 = \sum_{b_2,\ldots,b_n} \varphi(0, b_2, \ldots, b_n)$
$N_1 = \sum_{b_2,\ldots,b_n} \varphi(1, b_2, \ldots, b_n)$
	$N_0, N_1 \longrightarrow$
	$\longleftarrow a_1$
		if $N_0 + N_1 \neq N$ **then**
		  reject
		end if
		$a_1 \xleftarrow{R} \{0,1\}$

$N_{a_1,0} = \sum_{b_3,\ldots,b_n} \varphi(a_1, 0, b_3, \ldots, b_n)$
$N_{a_1,1} = \sum_{b_3,\ldots,b_n} \varphi(a_2, 1, b_2, \ldots, b_n)$
	$N_{a_1,0}, N_{a_1,1} \longrightarrow$
	$\longleftarrow a_2$
		if $N_{a_1,0} + N_{a_1,1} \neq N_{a_1}$ **then**
		  reject
		end if
		$a_2 \xleftarrow{R} \{0,1\}$

$\vdots$		

$N_{a_1,\ldots,a_n} = \varphi(a_1, \ldots, a_n)$	$N_{a_1,\ldots,a_n} \longrightarrow$	**if** $N_{a_1,\ldots,a_n} = \varphi(a_1, \ldots, a_n)$ **then**
		  accept
		else
		  reject
		end if

Although this proof system wont work, we can get insight from analyzing it. Clearly this scheme has an efficient verifier and satisfies perfect completeness. Let's try to prove soundness: Assume φ has fewer than k satisfying assignments and let P^* be a cheating prover. Let $N^*, N_0^*, N_1^*, \ldots$ be the messages sent by P^* and let $N, N_0, N_1, \ldots$ be the correct messages (computing as in the description of the honest prover P given above). Now we can look at all the ways that the prover can be caught cheating:

- We know $N^* \neq N$ or else V would reject immediately.
- $\Longrightarrow N_0^* + N_1^* \neq N_0 + N_1$ or else V would reject.
- $\Longrightarrow$ With probability $\geq 1/2$ over the choice of a_1, $N_{a_1}^* \neq N_{a_1}$, because either $N_0^* \neq N_0$ or $N_1^* \neq N_1$, from the previous fact.

- $\Longrightarrow$ With probability $\geq 1/2$ over the choice of a_2, $N^*_{a_1,a_2} \neq N_{a_1,a_2}$.

$$\vdots$$

- $\Longrightarrow$ With probability $\geq 1/2$ over a_n, $N^*_{a_1,\ldots,a_n} \neq N_{a_1,\ldots,a_n} = \varphi(a_1,\ldots,a_n)$.

If we manage to choose $a_1,\ldots,a_n$ correctly in every round then we will catch P^*. But this will only occur with probability $1/2^n$, which is not sufficient. In retrospect, this scheme obviously can't work because we are only asking about the value of φ for one assignment, which can't help us prove that the number of satisfying assignments is large or small.

To prove the theorem, we will introduce a modified interactive proof system where the answers the prover sends are more robust.

Proof: (Theorem 3) Our strategy is to replace $\varphi : \{0,1\}^n \to \{0,1\}$ with a low-degree polynomial extension $\widetilde{\varphi} : \mathbb{Z}_p^n \to \mathbb{Z}_p$ that agrees with φ on boolean-valued input, but allow the random values a_i to range over all of $\mathbb{Z}_p$. We'll use the fact that two distinct low-degree polynomials can only agree on a few inputs to ensure that lies by the prover propagate with high probability (rather than only with probability $1/2$ as in the earlier protocol).

We can give a simple recursive algorithm to construct such an extension of φ using the following mapping:

- $\varphi(x) = x_i \longmapsto \widetilde{\varphi}(x) = x_i$
- $\varphi(x) = \neg\alpha(x) \longmapsto \widetilde{\varphi}(x) = 1 - \widetilde{\alpha}(x)$
- $\varphi(x) = \alpha(x) \wedge \beta(x) \longmapsto \widetilde{\varphi}(x) = \widetilde{\alpha}(x) \cdot \widetilde{\beta}(x)$

By inspection we can se that for every $b_1,\ldots,b_n \in \{0,1\}^n$, $\widetilde{\varphi}(b_1,\ldots,b_n) = \varphi(b_1,\ldots,b_n)$. Also, $\deg(\widetilde{\varphi}) \leq |\varphi|$, and $\widetilde{\varphi}$ can be evaluated on any input in $\mathbb{Z}_p^n$ in time $\text{poly}(|\varphi|, \log p)$. (The latter property is the advantage of this recursive arithmetization over the general low-degree extensions we saw last lecture (which required exponential summation to compute).)

The proof system can be modified in the following ways:

- In the first round P sends $N = \sum_{b_1,\ldots,b_n \in \{0,1\}^n} \widetilde{\varphi}(b_1,\ldots,b_n)$ and a prime $2^{|\varphi|} < p \leq 2^{|\varphi|+1}$.
- Instead of N_0 and N_1, P sends the polynomial $N_\epsilon(x) = \sum_{b_2,\ldots,b_n \in \{0,1\}^{n-1}} \widetilde{\varphi}(x, b_2,\ldots,b_n)$ (where all computation is in now in $\mathbb{Z}_p$). Note that $N_\epsilon(0) = N_0$ and $N_\epsilon(1) = N_1$.
- The verifier checks the primality of p, that $k \leq N \leq 2^n$, and that $N_\epsilon(0) + N_\epsilon(1) = N \bmod p$. Then the verifier chooses $a_1 \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$.
- In each round, after receiving $a_1,\ldots,a_k \in \mathbb{Z}_p$, P computes the polynomial

$$N_{a_1,\ldots,a_k}(x) = \sum_{b_{k+2},\ldots,b_n} \widetilde{\varphi}(a_1,\ldots,a_k,x,b_{k+2},\ldots,b_n)$$

  and sends it to V. V checks that $N_{a_1,\ldots,a_k}(0) + N_{a_1,\ldots,a_k}(1) = N_{a_1,\ldots,a_{k-1}}(a_k)$.
- At the end, V checks that $N_{a_1,\ldots,a_n}$ equals the constant $\widetilde{\varphi}(a_1,\ldots,a_n)$.

Again, efficiency and completeness are easy to verify. The proof of soundness is as follows: Assume φ has fewer than k satisfying assignments and let P^* be some cheating prover. Let $N^*, N^*_\epsilon(x), \ldots$ be the messages sent by P^* and let $N, N_\epsilon(x), \ldots$ be the correct messages.

- We know $N^* \neq N$ or else V would reject immediately.
- $\implies N^* \bmod p \neq N \bmod p$
- $\implies N^*_\epsilon(0) + N^*_\epsilon(1) \neq N_\epsilon(0) + N_\epsilon(1)$.
- $\implies N^*_\epsilon(x)$ and $N_\epsilon(x)$ are different polynomials of degree at most $|\varphi|$.
- $\implies$ With probability $\geq 1 - |\varphi|/p$ over the choice of $a_1 \xleftarrow{\text{R}} \mathbb{Z}_p$, $N^*_\epsilon(a_1) \neq N_\epsilon(a_1)$.

By the union bound, we conclude that the soundness error for the whole protocol is $\leq \frac{n|\varphi|}{p} = o(1)$ because we chose $p \geq 2^{|\varphi|}$. ■