CIS 511: Theory of Computation **Apr 6, 2017**

Lecture 21: Interactive Proof, Continued

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Interactive Proof (continued)

We were in the middle of proving that there is an interactive proof for $\#SAT$. **Proof:** Recall ϕ was our original formula over $x_1, \ldots, x_m$ and its arithmetization Φ, where $\phi(\vec{x}) = \Phi(\vec{x})$ if $\vec{x}$ is a binary vector/boolean assignment. We didn't say this last time, but the polynomial verifier can do this arithmetization.

The prover wants to convince the verifier that ϕ has k satisfying assignments. Equivalently, $\sum_{i=1}^{m} \sum_{v_i \in \{0,1\}} \phi(x_1, \ldots, x_k) = k$. If we replace ϕ with Φ, we should have the same thing, and the verifier can do just that.

Let's decompose the sum. We have $\sum_{v_2 \in \{0,1\}} \sum_{v_3 \in \{0,1\}} \cdots \sum_{v_m \in \{0,1\}} \Phi(z, x_2, x_3, \ldots, x_m) = F_1(z)$. There are 2^{m-1} possible assignments for the remaining $m-1$ variables. We therefore have a polynomial over our finite field $\mathcal{F}$ in z of degree no more than the number of times x_1 occurs, which is no more than the length of the input, n.

The prover sends $F_1(z)$ to the verifier. $F_1(0) + F_1(1)$ should equal k. The verifier can plug these in for z and check that the sum is equal to k. If not, the verifier rejects. The verifier then picks an element $r_1 \in \mathcal{F}$ at random, and asks the prover to prove that $F_1(r_1)$ has the value it is supposed to have.

The prover then looks at $F_2(r_1, z)$, which is the summation over $x_3 \ldots x_m \in \{0,1\}$ of $\Phi(r_1, z, x_3 \ldots)$. The prover sends the coefficients of $F_2(r_1, z)$ to the verifier. The verifier checks that $F_2(r_1, 0) + F_2(r_1, 1) = F_1(r_1)$. Then, it picks a random $r_2 \in \mathcal{F}$, and asks the prover to prove that $F_2(r_1, r_2)$ has the value it should. This procedure repeats until we get to F_m.

We now need to verify correctness. Suppose the prover is honest and $\#\phi = k$, then the prover will convince the verifier of this fact. Suppose the prover may be dishonest. We will show that with high probability, the verifier will catch a lie. If the prover $\tilde{P}$ claims $\#\phi\tilde{k} \neq k$, let $\tilde{F_1}(z)$ be the polynomial that $\tilde{P}$ sends the verifier in the first round. The correct polynomial is $F_1(z)$. Could $\tilde{F_1}(z) = F_1(z)$? No! Because then $\tilde{F_1}(0) + \tilde{F_1}(1) = F_1(0) + F_1(1) = k$. $\tilde{F_1}(z)$ must be different, otherwise $\tilde{P}$ would be caught immediately.

Then, $\tilde{F_1}(z) - F_1(z) \not\equiv 0$. This difference polynomial has degree less than or equal to n. The verifier picks r_1 at random. Then, at most n values for r_1 will be a root of $\tilde{F_1}(z) - F_1(z)$. Therefore, the probability that r_1 is a root is at most $\frac{n}{|\mathcal{F}|}$. If r_1 is not a root of this difference, then $\tilde{F_1}(r_1) \neq F_1(r_1)$, and the prover is still trying to prove something false. At any round, the prover is only off the hook if the verifier happens to pick a root of this difference for r_i.

At the end, $\tilde{P}$ claims $\tilde{F}_m(r_1, r_2, \ldots, r_m) = \tilde{k}$, but this is equal to $\Phi(r_1, r_2, \ldots, r_m)$, which the verifier can check itself, catching a lie. What is the probability that a lying prover does not get caught? The probability that a prover escapes detection is no more than $\frac{n}{|\mathcal{F}|}$. Therefore, the probability that the verifier catches a dishonest prover is $1 - \frac{n}{|\mathcal{F}|}$.

As long as we pick a field of size at least $3n^2$, we have a valid interactive proof.

■

To complete the proof that $IP = PSPACE$, we need to show $PSPACE \subseteq IP$ by taking a $PSPACE$-Complete language and showing it has an interactive proof. We can do this for TQBF. The structure of the proof will take a formula $\psi = q_1 x_1 q_2 x_2 \dots q_m x_m \phi(x_1, x_2 \dots x_m)$ and perform arithmetization to strip variables off and reduce the problem. For the quantifiers, we can replace the $FOR\ ALL$s by ANDing two formulas and the $THERE\ EXISTS$ by ORing two formulas. The degrees of this polynomial will grow quickly by this process, so it may take exponential space to write the polynomial down. Therefore, the verifier can't compute the arithmetization. We fix this by replacing high degrees of variables with lower degrees, because they will still agree on $x = 0, 1$. This is subtle and a little tricky, but doable.

Zero Knowledge Proofs

Interactive proof is important in cryptography. We are interested in cases where Alice uses a 'secret' to perform a task to convince Bob that the she knows the secret without revealing it to Bob.

We want to show that zero-knowledge proofs are sound (a dishonest prover gets caught), complete (if both parties are honest, the prover can convince the verifier), and zero-knowledge (an honest verifier doesn't reveal the secret).

Let's suppose we have a graph $G = (V, E)$. The prover wants to convince V in zero knowledge that G is 3-colorable, which it knows. If we didn't have the zero-knowledge requirement, we could just treat this as an NP language and be done in one step.

Here's the interactive proof. At each round, the prover creates a random graph H isomorphic to G and permutes the colors. The prover correctly 3-colors H and hides the colors and the edges. The prover could be lying in a couple of ways: H could not be isomorphic to G or the coloring it picked is not a valid 3-coloring. The verifier knows the original graph G and the three colors that should be used.

The verifier is allowed to make one of two challenges at each round. It can ask the prover to prove that the presented graph is isomorphic to G. This can be done by revealing the edges and giving a short proof of isomorphism. The second challenge is that the verifier can pick any two vertices and ask the prover to show that they are properly colored. This can be done by revealing at most the edge between them and the respective coloring. If the prover succeeds in showing something valid, then we move to the next round. Otherwise, the verifier catches a lie and rejects.

What is the probability the cheating prover gets caught? If the graph is not isomorphic, then the prover gets caught with probability $\frac{1}{2}$. If the prover is lying about the colorabilty of the graph, then there is some pair of vertices colored illegally. The verifier will catch this with probability $O(\frac{1}{n^2})$. Therefore, in n^3 rounds, the probability of failing to catch a lying prover is $O(e^{-n})$.

Where does zero-knowledge come in? An IP is zero-knowledge if the transcript could have been generated without the verifier actually looking at the graph. Since challenges and vertex pairs are chosen at random, it could commit, a priori, to a (secret) order of challenges, so the verifier doesn't really learn anything from the prover. Since the verifier can simulate the protocol, it is zero-knowledge.