6.841 Advanced Complexity Theory April 8, 2009

Lecture 17

Lecturer: Madhu Sudan *Scribe: Jean Yang*

1 Overview

Last lecture we showed $PERM \in IP \Rightarrow \#P \in IP$ by constructing an IP protocol involving verifying a polynomial using a curve. In this lecture we review the method for showing $PERM \in IP$ and apply the method first for showing $\#SAT \in IP$ and then for showing $PSPACE \subseteq IP$.

In the second half of lecture we discussed "knowledge," its definition, and its relationship to crytography and complexity.

1.1 Administrative notes

- No lecture on Monday.
- Madhu will be away–Swastik will lecture on PCP.

2 Review from last time: $\#P \in IP$

We showed that $PERM \in IP$ problem using a polynomial construction sequence. A cool part is that this method would work for *any* problem that we can deconstruct into a sequence of polynomials, so we can also apply this method to showing $\#SAT \in IP$ and, eventually, $PSPACE \subseteq IP$.

2.1 Polynomial construction sequence

Recall from last lecture that we can compute the permanent in field $\mathbb{F}$ using the sequence of n polynomials $P_1, P_2, \ldots, P_{\ell-1}, P_\ell$ where P_i is the permanent of an $i \times i$ matrix. We have the following properties:

Each polynomial of degree	$\leq$	d
Each polynomial has number of variables	$\leq$	m
P_0 is computable in time	$\leq$	t
P_i is computable in time t with oracle for P_{i-1} with # calls	$\leq$	$w = n$

We get the last property from the downward self-reducibility of the permanent: $P_n(M) = \sum_{i=1}^{n} m_{1i} P_{n-1}(M_i)$.

Given $\overline{a} = (a_1, \ldots, a_m) \in \mathbb{F}^m$ and $b \in \mathbb{F}$, we can prove interactively that $P_\ell(\overline{a}) = b$ in time polynomial in $\ell, d, \mathbb{F}, m, t, w)$, provided that $|\mathbb{F}|$ is sufficiently large.

We showed in the previous lecture a method for verifying $PERM$ by running a curve through the space and then asking about a random point on the curve. We can work backwards from P_{n-1}.

2.2 Typical phase of interaction

We show how to verify the question "$P_i(a^{(i)}) = b^{(i)}$?" for polynomial P_i in a sequence of polynomials.

To verify $P_n(M) = a$, we can compute the w inputs $M_1, M_2, \ldots, M_w$ to use the oracle for P_{n-1} so that we can easily compute $P_n(M)$ from $P_{n-1}(M) \ldots P_{n-1}(M_w)$. For verification purposes we use a curve $C : \mathbb{Z}_p \to \mathbb{Z}_p^v$ (where $\mathbb{Z}_p^v$ is our input space) such that $C(1) = M_1, \ldots, C(w) = M_w$. Note $\deg(C) \leq w$ (actually $\deg(C) = w - 1$).

The interaction goes as follows:

- We ask the prover "$P_{n-1} \circ C =$?"

- The prover comes back with an answer h with degree $\leq d \cdot w$.
- The verifier verifies that "$P_n(M) = a$" is consistent with $P_{n-1}(M_i) = h(i)$ by picking a t at random and verifies $P_{n-1}(C(t)) = h(t)$.

The interaction looks like this:

"$P_i(a^{(i)}) = b^{(i)}$?"

Verifier		Prover
	$\xleftarrow{C,h}$	Compute $v_1 \dots v_w$ s.t. $P_i(a^{(i)})$ can be computed from $P_{i-1}(v_1 \dots v_2)$. Compute curve C such that $C(j) = v_j$ and $h \leftarrow P_{i-1}(C(t))$.
Verify $C(i) = v_i$.	$\xrightarrow{t_0}$	
Verify $b^{(i)} = f_i(h(1) \dots h(w))$.		
Pick $t_0 \in \mathbb{F}$ at random,		
$a^{(i-1)} = C(t_0); b^{(i-1)} = b(t_0)$.		

The notes for last lecture explain why we can verify with high probability. We can keep walking backwards and verifying until we get to the base case, at which point we'll know the answer with high probability.

2.3 IP protocol for $\#SAT$

Though we get $\#P \subseteq IP$ because $PERM$ is $\#P$-complete, we can show why this is quite clearly with a similar IP protocol for $\#SAT$. We will also use a similar method for showing $PSPACE \subseteq IP$.

Theorem 1 *Let $\#3SAT$ be the number of satisfying clauses of formulae of the form $\phi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, where $c_j = x_{i_1} \vee x_{i_2} \vee x_{i_3}$.*

$\#3SAT \in IP$.

Proof We can write a $\#3SAT$ formula naturally as a set of polynomials with the desired polynomial parameters.

We want a clause c_i to be satisfied $\iff$ the corresponding polynomial expression is equal to 1, and we want a formula to be satisfied $\iff$ every clause is satisified. To convert ϕ to a polynomial, we have:

$$\begin{aligned} x_1 \dots x_n &\rightarrow y_1 \dots y_n, \text{ where } x_i \text{ is true} \iff y_i = 1 \\ \overline{x_i} &\rightarrow 1 - y_i \\ c_i &\rightarrow 1 - (1 - y_{i_1})(1 - y_{i_2})y_{1_3} = \hat{p}_j \\ \phi &\rightarrow \Pi_j \hat{p}_j = p(y_1, \dots, y_n) \end{aligned}$$

To compute ϕ, we can go from many variables to few variables by computing each polynomial from the value of the previous polynomial, looking at only 2 places:

$$\begin{aligned} \hat{p}_0 &= \hat{p} \\ \hat{p}_1 &= \hat{p}_0(y_1, \dots, y_{n-1}, 0) + \hat{p}_0(y_1, \dots, y_{n-1}, 1) \\ &\vdots \\ \hat{p}_i &= \hat{p}_{i-1}(y_1, \dots, y_{n-i}, 0) + \hat{p}_{i-1}(y_1, \dots, y_{n-i}, 1) \\ &\vdots \\ \hat{p}_n() &= \hat{p}_{n-1}(0) + \hat{p}_{n-1}(1) \end{aligned}$$

For an interactive proof for the result of the constant function $\hat{p}_n$, we want walk backwards and ask about the previous function at two points, drawing a curve (line) between them. We know p_0 is of low degree, and summation does not increase the degree of the interactive proof.

A surprising thing is that we can pick a random value of t–everything works when we set t as arbitrary integers over some field. ■

3 $PSPACE \subseteq IP$

Now we show a $PSPACE$ computation where we want to compute the last step in a sequence of polynomials. We show how to take a generic $PSPACE$ computation and produce an IP problem by providing

1. a polynomial construction sequence that, given starting and ending configurations, can determine if we can get from the starting to ending configuration in a specified number of steps, and
2. a polynomial construction to compute each polynomial in the above sequence while meeting the requirements.

Theorem 2 *$PSPACE \subseteq IP$.*

Proof Fix a $PSPACE$ language L and machine M. We want to know if, starting from state c_0, we reach state c_f in 2^s steps, where we use $O(s)$ space.

We want to build a function $P_s(x, y)$ such that

$$\begin{aligned} p_s(x = (x1, \ldots, x_n), y = (x_1, \ldots, x_n)) \quad &= 1 \text{ if } x \to y \text{ in } 2^s \text{ steps} \\ &= 0 \text{ if } x, y \in \{0,1\}^s \text{ otherwise} \end{aligned}$$

To relate this polynomial to something smaller, we can look at the midpoint 2^{s-1}:

$$\begin{aligned} p_s(x, z) \quad &= 1 \text{ if } x \to z \text{ in } 2^{s-1} \text{ steps} \\ &= 0 \text{ otherwise} \end{aligned}$$

Note that there is exactly one such z, so we can just sum over the possible z's:

$p_s(x, y) = \sum_{z \in \{0,1\}^s} P_{s-1}(x, z) \circ P_{s-1}(z, y)$

We have $p_0(x, y), \ldots, p_s(x, y)$, where $p_0(x, y)$ is whether we can get from $x \to y$ in one step. The final question is "$p(c_0, c_f) = 1$?"

p_0 is a polynomial of low degree ($O(s)$ because we are looking at the product of $O(s)$ $\wedge$ conditions), exists, and is computable. Each expression of the form $p_{s-1}(x, z) \circ p_{s-1}(z, y)$ increases the degree of z, but se sum over all z's, so the degree does not go up in each iteration! Now we have left to solve the problem of needing the values of exponentially many things in order to take the sum.

We want to get to p_n with the polynomial sequence $p_0, p_1, \ldots p_{n-1}, p_n$, but to do so we have to resolve the issue that p_i needs 2^s values of p_{i-1}. We can use the $\#SAT$ trick and break up the sum into another series of polynomials. To compute $p_{n-1}(x, z)$, we can define a new sequence of polynomials

$$\begin{aligned} Q_s^0(x, z, y) \quad &= p_{n-1}(x, z) \circ p_{n-1}(z, y) \\ &\vdots \\ Q_s^i(x, z, y) \quad &= Q_s^{i-1}(x, (z_1, \ldots, z_{s-i}, 0), y) + Q_s^{i-1}(x, (z_1, \ldots, z_{s-i}, 1), y) \end{aligned}$$

At the end, we have $P_s = Q_s^s$. Let c be the degree in each variable. We have:

$$\begin{aligned} \text{degree} \quad &\leq \quad 4 \cdot c \cdot s \\ \text{length} \quad &\leq \quad O(s^2) \\ \text{width} \quad &= \quad 2 \\ \text{time} \quad &= \quad O(s) \\ \#\text{ variables} \quad &\leq \quad 3s \end{aligned}$$

Thus our sequence is good over every large field. ■

3.1 Notes on the implications and importance of this result

Previously, the class IP was not thought to be this powerful: there was an assumption that the number of rounds of interaction would collapse to a constant. The IP problem for the permanent was the first very counterintuitive proof and triggered a series of other proofs.

4 (Zero) knowledge

In class we introduced the concept of zero knowledge with the graph isomorphism example and then defined it more formally using the concept of replacing the verifier with a simulator. The professor's notes contains more information about information theory and results in zero knowledge which are not included here.

4.1 Definition by example

Consider the graph isomorphism problem: $G \simeq H$, we are asking

$\exists\pi$: $V(G) \to V(H)$
π one-to-one
such that $x \leftrightarrow y \iff \pi(x) \leftrightarrow \pi(y)$ in H

For a zero knowledge proof, we want the following properties with respect to the verifier V:

- **Completeness.** If $G_0 \simeq G_1$ then V must accept w.h.p.
- **Soundness.** If $G_0 \not\simeq G_1$ then V must reject w.h.p.
- **Zero knowledge.** If $G_0 \simeq G_1$ then V must not know isomorphism or learn anything other than this fact.

Consider the following interactive proof [Goldreich, Micali, Wigderson]:

	(G_0, G_1)	
Verifier		Prover
	$\xleftarrow{H}$	$b \in_R \{0,1\}$
		$\pi : V(G_b) \to V(G_b)$, π one-to-one
		randomly let $H \leftarrow \pi(G_b)$
choose $b' \in \{0,1\}$	$\xrightarrow{b'}$	
	$\xleftarrow{\pi'}$	$\pi' \leftarrow \pi$ if $b' = b$
		$\pi' \leftarrow \pi \circ \pi_0$ if $b = 0$, $b' = 1$
verify $H = \pi'(G_{b'})$		$\pi' \leftarrow \pi \circ \pi_{-1}$ if $b = 1$, $b' = 0$

If $G_0 \simeq H$, $H \simeq G_1$, then the prover can easily provide a permutation π' such that $\pi'(G_{b'}) = H$; otherwise the prover must count on the verifier choosing $b' = b$. The prover will try to cheat when one of these isomorphisms is missing–in this case the prover has probability $\frac{1}{2}$ of getting the correct permutation. Note that we can repeat the interaction, but the prover will have a new H.

Why did we do this proof when simply providing the isomorphism could have sufficed? The motivation is from crytography: we want to be able to convince others of facts iwthout revealing additional information.

4.2 Definition of zero knowledge

Anecdote. Micali put problem on problem set to prove that you don't know anything at the end of the interaction. Cook aid he didn't know how to prove he can't know something. This led Micali, Goldwasser, and Rackoff to come up with a set of definitions that allow you to show this.

We want to be able to formally define the notion "You learn *nothing* from the fact that (G_0, G_1) are isomorphic."

We use the concept of a "simulation." At the end of an interaction, we have a nondeterministic transcript of the interaction. If the verifier can simulate the interaction by sampling from some distribution without the prover, then we have a zero knowledge proof.

1. Fix verifier's random coins R.
2. Transcript is still random variable with distribution D_R.

If the verifier can sample from D_R on its own, then verifier gains no knwoledge from prover. The "simulator" samples from D_R.

Simulator for graph isomorphism:

Verifier		Simulator
	$\xrightarrow{b}$	pick $\pi \in S_n$
	$\xleftarrow{H}$	$H \leftarrow \pi(G_b)$
	$\xleftarrow{\pi}$	

We get exactly the same distribution as with a prover! Note, however, that in the graph isomorphism proof, the fact that H is fixed before the verifier responds is important for *convincing* the verifier, but it does not change the distribution of transcripts. The sequence only has convincing power–important for soundness.

4.3 Complexity theory of knowledge

We have the following classes of zero knowldge problems, where $PZK \subseteq SZK \subseteq CZK$:

- **Perfect zero knowledge.** Simulator can sample *exactly* from same distribution as interaction.
- **Statistical zero knowledge.** Simulator can sample from distribution that is 2^{-n} close to the real interaction. Simulator produces $D'_R \approx_\epsilon D_R$, so we have $\sum_n |D'(n) - D(n)| \leq 2\epsilon$. Equivalently $\forall$ tests $T = \{0,1\}^n \to \{0,1\}$, we have $|Pr_{n \in D_R}[T(n) = 1] - Pr_{n \in D'_R}[T(n) = 1]| \leq \epsilon$.
- **Computational zero knowledge.** Polynomial time does not suffice to distinguish simulated distribution from real one. The simulator produces a D''_R such that $\forall$ poly-time algorithms A, $|Pr_{n \in D_R}[A(x) = 1] = Pr_{n \in D''_R}[A(n) = 1]| \leq \epsilon$.

  Which is relevant where?

"$x \in L$"	in	PZK?	
		SZK?	[complexity]
		CZK?	[cryptography]

4.4 Some notes on SZK

SZK is closed under complementation: $L \in SZK \iff \overline{L} \in SZK$. This is surprising and nice because it gives us a complete promise problem.

Let $x_1, \ldots x_n$ be inputs and $y_1, \ldots, y_m$ be outputs. $C(x)$ when $x_1, \ldots, x_n \in \{0,1\}$ uniform, independent, is *some* distribution on m bit strings. (???)

Let $||C - D||$ be the distance between distributions. We have
$||C - D|| = \frac{1}{2} \sum_{y \in \{0,1\}^m} |Pr_x[C(x) = y] - Pr_x[D(x) = y]|$

We get the following complete promise problem for SZK:

$SD_{YES} = \{(C, D) \mid ||C - D|| \geq \frac{2}{3}\}$
$SD_{NO} = \{(C, D) \mid ||C - D|| \leq \frac{1}{3}\}$

The class SZK is worth examining further. We also have $SZK \subseteq AM$.