Lecture 13

Instructor: Madhu Sudan *Scribe: Chun-Yun Hsiao*

Today:

- **#P** $\subseteq$ **IP**
- **PSPACE** = **IP**

In this lecture, we will show that **PSPACE** $\subseteq$ **IP**; together with **PSPACE** $\supseteq$ **IP**, proved in last lecture, we conclude that **PSPACE** = **IP**. We proceed by first showing #**P** $\subseteq$ **IP**, and then generalize the proof to showing **PSPACE** $\subseteq$ **IP**.

1 #P is in IP

Let's begin by recalling what #**P** and **IP** are:

- #**P**: class of functions that count the number of accepting paths of a poly-time NTM. It suffices to consider a #**P**-complete problem. For practical purposes that will become clear later, we choose the problem $\#SAT$. That is, given a 3CNF, determine the number of satisfying truth assignments it has.
- **IP**: class of languages L, where "$x \in L$" has an interactive proof verifiable by a probabilistic poly-time TM.

Self-Reducibility Our goal is to show that $\#SAT$ has an interactive proof. More precisely, given a 3CNF formula ϕ and a number A, the all-powerful prover wants to convince the poly-time verifier that ϕ has exact A satisfying truth assignments. The idea is to exploit the self-reducibility of SAT. Let ϕ_0 be the formula ϕ with its first variable set to 0, i.e., $\phi_0 \triangleq \phi(x_1 = 0)$; similarly $\phi_1 \triangleq \phi(x_1 = 1)$. Suppose that we (the verifier) are convinced that the number of truth assignments of ϕ_0 and ϕ_1 are A_0 and A_1 respectively, then all we need to do is to check if $A = A_0 + A_1$. To make sure that A_0 and A_1 are the alleged numbers, we need to check that $A_0 = A_{00} + A_{01}$ and $A_1 = A_{10} + A_{11}$,[1] and recursively until all the variables are assigned with a Boolean value so that we can evaluate the A's ourselves.

Expanding the Tree? The problem is that every time we reduce one variable, we double the number of equations to be verified. To get around with this exponential growth, the prover, instead of giving A_0 and A_1, will provide us a function Q_1 such that $Q_1(0)$ and $Q_1(1)$ (are supposed to) represent the number of satisfying truth assignments of ϕ_0 and ϕ_1, respectively. Then the prover will provide another function Q_2 to convince us that Q_1 is the "right" function, and Q_3 for the validity of Q_2 and so on. The protocol proceeds until finally we can verify the validity of Q_n by ourselves. More precisely, we know that[2]

$$\begin{aligned} A &= \sum_{(x_1,\ldots,x_n)\in\{0,1\}^n} \phi(x_1, x_2, \ldots, x_n) \\ &= \sum_{x_1\in\{0,1\}} \sum_{x_2\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} \phi(x_1, x_2, \ldots, x_n). \end{aligned}$$

[1] A_{st} is defined analogously when the first two variables are assigned to s and t.

[2] Here the output of ϕ is viewed as an integer.

Let's define functions Q's without worrying how to represent them (efficiently) first. Naturally,

$$Q_1(x_1) \triangleq \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \phi(x_1, x_2, \ldots, x_n),$$

since this way $Q_1(0) + Q_1(1)$ is equal to A. We can generalize the definition to Q_i, for each i,

$$Q_i(x_1, x_2, \ldots, x_i) \triangleq \sum_{x_{i+1} \in \{0,1\}} \sum_{x_{i+2} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \phi(x_1, x_2, \ldots, x_n).$$

An ideal way of representing the Q's is by polynomials. For the polynomial representation to be meaningful, we require that they agree with our definition on binary inputs. To this end, we need an arithmetic way of looking at $\#SAT$.

Arithmetization Consider the following transformation from Boolean formulae to arithmetic polynomials.

Boolean		Arithmetic
$\bar{x_1}$	$\rightarrow$	$1 - x_1$
$C_j = (x_1 \vee \bar{x_2} \vee x_3)$	$\rightarrow$	$P_j = 1 - (1 - x_1)x_2(1 - x_3)$
$\phi(x_1, x_2, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$	$\rightarrow$	$P(x_1, x_2, \ldots, x_n) = \prod_{j=1}^{m} P_j(x_1, x_2, \ldots, x_n)$

It is easy to see that the polynomial P agrees with the formula ϕ on binary inputs $\{0,1\}^n$. And we can "redefine" the function Q_i in terms of P, namely for each i,

$$Q_i(x_1, x_2, \ldots, x_i) \triangleq \sum_{x_{i+1} \in \{0,1\}} \sum_{x_{i+2} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} P(x_1, x_2, \ldots, x_n).$$

Note that now each Q_i is indeed a polynomial. Furthermore, they are all of degree at most $3m$; "efficient" representation exists. The question is how do we verify these Q_i are derived honestly from $P(x_1, x_2, \ldots, x_n)$? Recall, from the definition, that the polynomial $Q_i(x_1, x_2, \ldots, x_i)$ should be identical to $Q_{i+1}(x_1, x_2, \ldots, x_i, 0) + Q_{i+1}(x_1, x_2, \ldots, x_i, 1)$. If we can verify their equality on a random input, they are identical with good probability. For our purposes, all arithemic operations will be done modulo some large prime p. Here is the protocol.

$\swarrow$	(ϕ, A)	$\searrow$
Verifier		**Prover**
REJECT if p not prime	$\xleftarrow{p}$	pick prime $p \in \{0,1\}^{n+1}$
REJECT if $h_1(0) + h_1(1) \neq A$	$\xleftarrow{h_1(\cdot)}$	send coefficients of $Q_1(\cdot)$
wonder "$h_1(\cdot) \stackrel{?}{=} Q_1(\cdot)$"		
challenge $\alpha_1 \in_{\mathrm{R}} \mathbb{Z}_p$	$\xrightarrow{\alpha_1}$	show "$A_1 \triangleq h_1(\alpha_1) = Q_1(\alpha_1)$"
REJECT if $h_2(0) + h_2(1) \neq A_1$	$\xleftarrow{h_2(\cdot)}$	send coefficients of $Q_2(\alpha_1, \cdot)$
wonder "$h_2(\cdot) \stackrel{?}{=} Q_2(\alpha_1, \cdot)$"		
challenge $\alpha_2 \in_{\mathrm{R}} \mathbb{Z}_p$	$\xrightarrow{\alpha_2}$	show "$A_2 \triangleq h_2(\alpha_2) = Q_2(\alpha_1, \alpha_2)$"
REJECT if $h_i(0) + h_i(1) \neq A_{i-1}$	$\xleftarrow{h_i(\cdot)}$	send $Q_i(\alpha_1, \ldots, \alpha_{i-1}, \cdot)$
wonder "$h_i(\cdot) \stackrel{?}{=} Q_i(\alpha_1, \ldots, \alpha_{i-1}, \cdot)$"		
challenge $\alpha_i \in_{\mathrm{R}} \mathbb{Z}_p$	$\xrightarrow{\alpha_i}$	show "$A_i \triangleq h_i(\alpha_i) = Q_i(\alpha_1, \ldots, \alpha_i)$"
REJECT if $h_n(0) + h_n(1) \neq A_{n-1}$	$\xleftarrow{h_n(\cdot)}$	send $Q_n(\alpha_1, \ldots, \alpha_{n-1}, \cdot)$
$\alpha_n \in_{\mathrm{R}} \mathbb{Z}_p$		
REJECT if $h_n(\alpha_n) \neq Q_n(\alpha_1, \alpha_2, \ldots, \alpha_n)$		
ACCEPT		

Note that $Q_n(\alpha_1, \alpha_2, \ldots, \alpha_n) = P(\alpha_1, \alpha_2, \ldots, \alpha_n)$ can be computed by ourselves.

Completeness If indeed ϕ has A satisfying truth assignments, the honest prover just follows the protocol and sends the correct h_i, which is $Q_i(\alpha_1, \ldots, \alpha_{i-1}, \cdot)$. The verifier will accept with probability one.

Soundness Suppose that ϕ has $A' \neq A$ satisfying truth assignments. Inductively, if $Q_{i-1}(\alpha_1, \ldots, \alpha_{i-1}) \neq A_{i-1}$, then

$$h_i(0) + h_i(1) = A_{i-1} \neq Q_{i-1}(\alpha_1, \ldots, \alpha_{i-1}) = Q_i(\alpha_1, \ldots, \alpha_{i-1}, 0) + Q_i(\alpha_1, \ldots, \alpha_{i-1}, 1).$$

The first equality is assured by verifier's check, otherwise it would reject immediately. The last equality is by the definition of Q and the inequality is the inductive assumption. So we know that $h_i(\cdot) \neq Q_i(\alpha_1, \ldots, \alpha_{i-1}, \cdot)$; with probability $1 - \frac{3m}{p}$, α_i is not a rooot of $h_i(\cdot) - Q_i(\alpha_1, \ldots, \alpha_i - 1, \cdot)$. Thus $h_i(\alpha_i) \neq Q_i(\alpha_1, \ldots, \alpha_i)$. By union bound, the soundness is at least $1 - \frac{3mn}{p}$.

Remark Notice that the verifier only tosses *public* coins. This is another "indication" that **IP** is no more powerful than Athur-Merlin game.

2 PSPACE = IP

This result is somewhat surprising, in the following sense.

- We don't know (expect) that **IP** is closed under complement.
- There exists an oracle O such that $\mathbf{IP}^O \not\supseteq (\mathbf{\Sigma}_i^P)^O$.

Abstract of the Proof Note that in last section the proof didn't use any specific feature of #**P**. The key idea is only the downward self-reducibility. Let's look at the proof abstractly and see if it could lead us to showing **PSPACE** $\subseteq$ **IP**.[3]

1. $Q_0, Q_1, \ldots, Q_n$ is a sequence of low-degree polynomials.
2. $Q_0() = A$.
3. Q_n is a polynomial we can evaluate on any input by ourselves.
4. Q_i can be computed in poly-time, give non-adaptive oracle queries to Q_{i+1}.
   - For example, $Q_i(\alpha_1, \ldots, \alpha_i) = Q_{i+1}(\alpha_1, \ldots, \alpha_i, 0) + Q_{i+1}(\alpha_1, \ldots, \alpha_i, 1)$.
   - In general, $Q_i(\mathbf{y})$ can be computed from $Q_{i+1}(\mathbf{y}_1), Q_{i+1}(\mathbf{y}_2), \ldots, Q_{i+1}(\mathbf{y}_\ell)$, where $\mathbf{y}_j$ can be computed from $\mathbf{y}$.

Condition 4 is the most important one that saves us from exponential fan-out. Indeed, if we are only concerned about Q_{i+1} on $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$, we may focus on $Q_{i+1}|_C$, where C is a curve passing through $\mathbf{y}_1, \ldots, \mathbf{y}_\ell \in \mathbb{Z}_p^n$.

Let us formalize the above discussion. A curve C is a function (polynomial) from $\mathbb{Z}_p$ to $\mathbb{Z}_p^n$.

$$C = (C_1, C_2, \ldots, C_n), \text{ where each } C_i : \mathbb{Z}_p \to \mathbb{Z}_p \text{ is a polynomial.}$$

The degree of C is the maximum degree of C_i, i.e., $\max_i\{\deg(C_i)\}$. Here are some useful factoids.

- Given $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\ell \in \mathbb{Z}_p^n$, there exists degree $\ell - 1$ curve C such that $C(i) = \mathbf{y}_i$, $\forall i = 1, 2, \ldots, \ell$.

[3] In fact every self-reducible language is in **PSPACE** and as we shall see in next lecture that every language in **PSPACE** is self-reducible.

- Let $Q : \mathbb{Z}_p^n \to \mathbb{Z}_p$ be degree d polynomial, and $C : \mathbb{Z}_p \to \mathbb{Z}_p^n$ be degree ℓ polynomial. Then $Q|_C : \mathbb{Z}_p \to \mathbb{Z}_p$ is a polynomial of degree at most $d\ell$. We can write $Q|_C(t) = Q(C(t))$.

In general at the i^{th} round, we want to verify the validity of $h_i(\cdot)$, which is received in the previous round. We pick a random $\mathbf{y} \in \mathbb{Z}_p^n$, compute the curve C and send it to the prover. The prover respond with h_{i+1} which is supposed to be $Q_{i+1}|_C$. Since C passes through $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\ell$, we are able to compute $h_i(\mathbf{y})$ and notice any inconsistency so far, with good probability.

It turns that the curve C doesn't need to be complicated at all; straight lines will suffice for our purposes to show $\mathbf{PSPACE} \subseteq \mathbf{IP}$. We will elaborate more in the next lecture.

References

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *FOCS* 1990: 2-10.

[Sha90] Adi Shamir. IP=PSPACE. *FOCS* 1990: 11-15.

[Sud02] Madhu Sudan. 6.841/18.405J: Advanced Complexity Theory, Lecture 14, 2002. `http://theory.lcs.mit.edu/~madhu/ST02/scribe/lect14.ps`.