E0 224: Computational Complexity Theory **Dept of CSA**
Fall 2014 **Indian Institute of Science**

Lecture 23: Nov 3,2014

Lecturer: Chandan Saha <chandan@csa.iisc.ernet.in > *Scribes: Mohd Aqil,Datta Krupa R*

In this lecture we define the language ($\#SAT_D$) and prove that ($\#SAT_D$) $\in$ ***IP*** .This will help us in the next lecture to prove that **IP = PSPACE** [SHAM90, LFKN90] an amazing result that showed that when both randomization and interaction are allowed, the proofs that can be verified in polynomial time are exactly those proofs that can be generated with polynomial space.

Before Shamir proved this in 1990 (building upon the work of [LFKN90]) many researchers believed that **IP** was strict subset of **PSPACE** .This intuition was based on the following facts.It was known that interaction alone does not give us any language outside **NP**(**dIP** = **NP**) and randomization alone does not add significant power to computation(Researchers suspect **BPP** = **P**).So researchers believed that combining interaction and randomness could not give us a much powerful model.

Theorem 23.1 ***IP = PSPACE*** *[SHAM90, LFKN90]*

Definition 23.2 (IP[k]) *(recall)] Language $L \in IP[k]$ if there is a probabilistic poly time TM V (called the verifier) which interacts with a prover function P for k rounds, such that following holds,*
(completeness)

$$x \in L \Leftrightarrow \exists P \ s.t \ Pr_{randomness \ of \ V}\{Out_V(\langle V, P\rangle) = 1\} \geq \frac{2}{3}$$

(soundness)

$$x \notin L \Leftrightarrow \forall P \ s.t \ Pr_{randomness \ of \ V}\{Out_V(\langle V, P\rangle) = 1\} \leq \frac{1}{3}$$

Definition 23.3 (class IP) *(recall)*

$$IP = \cup_{c>0} IP[n^c]$$

We will prove theorem(23.1) in the next lecture.Here we will prove **IP** $\subseteq$ **PSPACE** and a weaker result **Co-NP** $\subseteq$ **IP**.

23.1 IP $\subseteq$ PSPACE

[ref3] **Proof:** Suppose L $\in$ **IP**.Then L $\in$ **IP**$[n^c]$ for some constant c.Assume $K = n^c$. Then we have some verifier V for L.For a fixed input x and fixed K, we observe that there are finitely many prover functions P.We choose the optimum prover P that maximizes the verifier's acceptance probability given input x.We denote this maximum probability by ρ_{opt}

$$\rho_{opt} = max_P Pr\{out_V < V, P > (x) = 1\} \tag{23.1}$$

if $\rho_{opt} \geq \frac{2}{3}$ then we have x $\in$ L. If $\rho_{opt} \leq \frac{1}{3}$ then we have x $\notin$ L.

To prove that **IP** $\subseteq$ **PSPACE** we will show that optimum prover runs in **PSPACE**. The crux of the proof lies in enumerating each possible communicating pattern between the verifier and prover for a fixed input x

.Suppose V runs in *p(n)* time where *p(n)* is some polynomial in n and $n =| x |$.The prover knows the verfier protocol but does not know the random bits selected by the verifier.However the length of random string can be atmost *p(n)* since verifier runs in time *p(n)* time.To generate the optimum response the prover uses recursion to simulate the verifier.We also note that the length of the response by the prover is also bounded by *p(n)* and the total number of interactive rounds is K , both of which are polynomials in n.To generate the optimum response at each round i, the prover simulates the action of verifier for each possible response over all possible random strings.Prover can easily simulate such a verifier by using recursion.We notice here that the depth of recusrion is atmost K which is polynomial in n and also each recusrive procedure uses polynomial amount of bits.Thus the entire procedure uses atmost polynomial space.By seeing the verifiers output for each possible response over all random strings, the prover chooses the response which causes the verifier to accept the maximum number of times.

■

23.2 Co-NP $\subseteq$ IP

To prove that **Co-NP** $\subseteq$ **IP** , we can show that $\overline{SAT} \in$ **IP** since it is *Co-NP Complete.* We will prove a slightly more general result $(\#SAT_D) \in$ **IP**.

Definition 23.4 $(\#SAT_D) = \{< \phi, K >$: *$\phi$ is a 3CNF formula and it has exactly K satisfying assignments*$\}$
We note that $(\#SAT_D)$ *contains* $\overline{SAT}$ *as a special case when K = 0.*

Definition 23.5 ***Arithmetization:****Given a boolean formula* $\phi(\overrightarrow{x})$ *define a polynomial* $P_\phi(\overrightarrow{x})$ *such that* $\forall \overrightarrow{x} \in \{0,1\}^n$, $\phi(\overrightarrow{x}) = \mathrm{P}(\overrightarrow{x})$. *Note that 0,1 can be thought of both as truth values and as elements of some finite field* $\mathbb{F}$.*For convenience of notation we use the same symbols to denote boolean variables as well as variables over field* $\mathbb{F}$.

We make the following observations:

- $x \wedge y$ is satisfiable iff x.y = 1 in the field $\mathbb{F}$ for x,y $\in \{0,1\}$.
- $\neg x$ is satisfiable iff 1-x = 1 in the field $\mathbb{F}$ for x $\in \{0,1\}$.
- $x \vee y$ is satisfiable iff 1-(1-x)(1-y) = 1 in the field $\mathbb{F}$ for x,y $\in \{0,1\}$. We have $x \vee y = \neg(\neg x \wedge \neg y)$ and the result follows.

Now given any 3-CNF formula $\phi(x_1, x_2, \ldots, x_n)$ with m clauses and n variables we can convert it into an equivalent polynomial.For any clause of size 3 we write an equivalent degree 3 polynomial as given in the following example.

$$x_i \vee \overline{x_j} \vee x_k \longleftrightarrow 1 - (1 - x_i)(x_j)(1 - x_k)$$

We denote the polynomial for the jth clause by $p_j(x_1, x_2, \ldots, x_n)$, for the sake of notation, even though p_j depends on atmost 3 variables.Multiplying the polynomials for each clause we obtain the multivariate polynomial $P_\phi(x_1, x_2, \ldots, x_n) = \Pi_{j \le m} p_j(x_1, x_2, \ldots, x_n)$ that evaluates to 1 on satisfying assignments of $\phi(x_1, x_2, \ldots, x_n)$ and 0 otherwise. The degree of $P_\phi(x_1, x_2, \ldots, x_n)$ is at most 3m since each $p_j(x_1, x_2, \ldots, x_n)$ is of degree at most 3. Also the size of $P_\phi(x_1, x_2, \ldots, x_n)$ is *O(m)* since P_ϕ is represented as a product of degree 3 polynomials without opening the parenthesis.

We observe here that once we have the corresponding polynomial for the boolean formula, we can evaluate $P_\phi(x_1, x_2, \ldots, x_n)$ at arbitrary values from the finite field $\mathbb{F}$ instead of just $\{0, 1\}$.

Theorem 23.6 $(\#SAT_D) \in$ **IP**

Proof: Given input $< \phi, K >$ where ϕ is a 3-CNF formula of n variables and m clauses, we use arithmetization to construct polynomial P_ϕ.Let $\#\phi$ denoting the number of satisfying assignments of ϕ. Then we have

$$\#\phi = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\phi(b_1, \ldots, b_n) \tag{23.2}$$

If the string $< \phi, K >$ is in $(\#SAT_D)$, then this sum $\#\phi$ is exactly K.We also note that the value of $\#\phi$ can be in the range $[0, 2^n]$.Now we describe the *sumcheck protocol* that allows us to verify that $\#\phi$ is indeed K as claimed by the prover.

Sumcheck Protocol First the prover sends a prime p to the verifier in the interval $(2^n, 2^{2n}]$.Verifier checks if p is a prime by any of the probabilistic or deterministic tests. All the computations will be performed modulo p in the sumcheck protocol i.e in the field $\mathbb{F} = \mathbb{F}_p$ of integers.

Let $g(X_1, X_2, \ldots, X_n)$ be an arbitrary degree d polynomial, then the claim of the prover is for $g = P_\phi$ (where the degree d of the polynomial is at most $3m$) .

$$K = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, \ldots, X_n) \tag{23.3}$$

where all the computations are in the field $\mathbb{F}_p$.We note the following:

- The polynomial $g(X_1, X_2, \ldots, X_n)$ should have a *poly(n)* representation so that the verifier can efficiently evaluate $g(X_1, X_2, \ldots, X_n)$ for any $X_1 = b_1, X_2 = b_2, \ldots, X_n = b_n$. This is true for $g = P_\phi$
- If we fix $X_2 = b_2, \ldots, X_n = b_n$ then $g(X_1, b_2, \ldots, b_n)$ is polynomial in one variable X_1 of degree d.
- The following is also a degree d polynomial

$$h(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2 \ldots, b_n) \tag{23.4}$$

- If the equation (23.3) is true then $h(0) + h(1) = K$.

The *sumcheck* protocol consists of the following steps by the verifier and the prover:

- **Verifier:** If $n = 1$, verify that $g(1) + g(0) = K$.If it is true then accept otherwise reject.If $n \geq 2$ then ask prover to send $h(X_1)$ as described above.
- **Prover:** Send some polynomial $s(X_1)$ (if the prover is honest then we have $s(X_1) = h(X_1)$) .
- **Verifier:** if $s(0)+s(1) \neq K$ reject else pick a random number a in $\mathbb{F}_p$ and fix $X_1 = a$.Let $g(a, X_2, \ldots, X_n)$ be the polynomial in $n - 1$ variables.Then recursively use the same protocol to verify

$$s(a) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2 \ldots, b_n) \tag{23.5}$$

Completeness: If the equation (23.3) is true then the verifier will accept with probability 1.

If the claim(23.3) is true then in each round i the prover sends $s(X_i) = h(X_i)$.Hence $s(0) + s(1) = K$ for each round and verifier will accept with probability 1 after n rounds.

Soundness: If the claim is false then the verifier will reject with probability at least $(1 - \frac{d}{p})^n$.

We prove the above result by induction on n.Assume the induction hypothesis for degree d polynomials in $n-1$ variables i.e verifier rejects the false claim with probability at least $(1 - \frac{d}{p})^{n-1}$.In the base case if $n = 1$ then verifier evaluates $g(0) + g(1)$ and rejects with probability 1 if it is not equal to K.

Suppose $k \neq \sum \cdots \sum P_\varphi(X)$ then a dishonest prover has to send $s(x_1) \neq h(x_1)$, otherwise it will get caught i.e.if the prover returns $h(X_1)$ for a false claim then verifier rejects immediately with probability 1.Suppose the dishonest prover sends a polynomial $s(X_1)$ different from $h(X_1)$ such that $s(0) + s(1) = K$. We then observe that $h(X_1) - s(X_1)$ is a non zero degree d polynomial and thus has at-most d roots .Hence there are at most d values a at which $s(a) = h(a)$.If a is picked randomly in the field $\mathbb{F}_p$ then

$$Pr_a[s(a) \neq h(a)] \geq 1 - \frac{d}{p} \tag{23.6}$$

If $s(a) \neq h(a)$ then the prover has to prove a false claim in the recursive step for a $n-1$ variable polynomial , by induction hypothesis the prover fails to prove this claim with probability at least $\geq (1 - \frac{d}{p})^{n-1}$.Hence

$$Pr[Vrejects] \geq \big(1 - \frac{d}{p}\big).\big(1 - \frac{d}{p}\big)^{n-1} = \big(1 - \frac{d}{p}\big)^n \tag{23.7}$$

■

Lemma 23.7 ***Co-NP* $\subseteq$ *IP***

Proof: We note that $(\#SAT_D)$ contains $\overline{SAT}$ as a special case when K = 0. Hence $\overline{SAT} \in$ **IP**. Since $\overline{SAT}$ is *Co-NP complete* therefore **Co-NP** $\subseteq$ **IP**. ■

References

[SHAM90] A. Shamir. IP = PSPACE. J. ACM 39(4): 869877 (1992). Preliminary version in FOCS 90.

[AROBAR09] S. ARORA and B. BARAK Computational Complexity: A Mordern Approach, Cambridge University Press, 2009

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

[ref3] http://cs.brown.edu/courses/gs019/papers/ip.pdf