E0 224 Computational Complexity Theory **Indian Institute of Science, Bangalore**
Fall 2014 **Department of Computer Science and Automation**

Lecture 24: Nov 5, 2014

Lecturer: Chandan Saha <chandan@csa.iisc.ernet.in> *Scribe: Sachin Kumar Srivastava*

24.1 IP = PSPACE

In the last class, we showed that #SAT $\in$ IP. The crucial step, there, was Arithmetization. We converted an instance of a 3 CNF formula to a polynomial.

$\varphi \rightarrow P_\varphi$ such that $\varphi(b_1, b_2, ..., b_n) = P_\varphi(b_1, b_2, ..., b_n)$, where $b_i \in \{0, 1\}$.

We have already argued, in the last class, that IP $\subseteq$ PSPACE. Now, we have to show that PSPACE $\subseteq$ IP. So, we only need to show that TQBF $\in$ IP as TQBF is PSPACE-complete.

TQBF = $\{$ All true quantified boolean formulae $\}$.

QBF : QBF has the form $Q_1x_1Q_2x_2Q_3x_3....Q_nx_n\varphi(x_1, x_2, x_3, ..., x_n)$, where each Q_i is either $\exists$ or $\forall$ and x_is are boolean variables.

Input : $y = Q_1x_1Q_2x_2Q_3x_3....Q_nx_n\varphi(x_1, x_2, x_3, ..., x_n)$.

First Observation : We can assume, without loss of generality, that $\varphi(x_1, x_2, x_3, ..., x_n)$ is a 3 CNF formula (because we can convert any SAT instance to an equivalent 3 CNF SAT instance in polynomial time). Let P_φ be the arithmetization of φ.

Second Observation : y is in TQBF if and only if $R_1x_1R_2x_2....R_nx_nP_\varphi(x_1, x_2, ..., x_n) \neq 0$,

where $R_i = \sum_{x_i \in \{0,1\}}$ if $Q_i = \exists$ and

$R_i = \Pi_{x_i \in \{0,1\}}$ if $Q_i = \forall$.

For example : $\exists x\varphi(x) \in TQBF \iff \sum_{x \in \{0,1\}} P_{\varphi(x)} \neq 0$ and $\forall x\varphi(x) \in TQBF \iff \Pi_{x \in \{0,1\}} P_{\varphi(x)} \neq 0$ (where degree of $P_\varphi \leq 3m$ and underlying field is $\mathbb{F}_p$ where p is sufficiently large i.e. $p \in [2^{2n^2}, 2^{3n^2}]$).

$\exists x_1 \forall x_2 \varphi(x_1, x_2) \in TQBF \iff \sum_{x_1 \in \{0,1\}} \Pi_{x_2 \in \{0,1\}} P_\varphi(x_1, x_2) \neq 0.$

We will prove by induction that :

$Q_1x_1Q_2x_2....Q_nx_n\varphi(x_1, x_2, x_3, ..., x_n) \in TQBF$ if and only if
$\left(\sum_{x_1 \in \{0,1\}} or \Pi_{x_1 \in \{0,1\}}\right)\left(\sum_{x_2 \in \{0,1\}} or \Pi_{x_2 \in \{0,1\}}\right)....\left(\sum_{x_n \in \{0,1\}} or \Pi_{x_n \in \{0,1\}}\right) P_\varphi(x_1, x_2, ..., x_n) = k$, where k is some non-zero positive integer and Q_i is either $\exists$ or $\forall$ and operator for quantification in polynomial for x_i depends upon Q_i, i.e. if Q_i is $\exists$, then operator will be $\sum_{x_i \in 0,1}$ and if Q_i is $\forall$, then operator will be $\Pi_{x_i \in \{0,1\}}$.

We can see that it is true for the base case when $n = 1$ i.e. if formula is of the form $\exists\varphi(x_1)$, then it belongs to TQBF iff for either $x_1 = 0, \varphi(x_1)$ is true or for $x_1 = 1, \varphi(x_1)$ is true, hence when it is converted to $\sum_{x_1 \in \{0,1\}} P_\varphi(x_1)$, $P_\varphi(x_1)$ will have a non-zero value iff either at $x_1 = 0, P_\varphi(x_1) > 0$ or at $x_1 = 1, P_\varphi(x_1) > 0$ or at both $P_\varphi(x_1) > 0$, in any case we will sum both values and hence it is true iff any of these values is greater than zero. On the other hand, if formula is of the form $\forall\varphi(x_1)$, then it belongs to TQBF iff for both $x_1 = 0, \varphi(x_1)$ is true and for $x_1 = 1, \varphi(x_1)$ is true, hence when it is converted to $\Pi_{x_1 \in \{0,1\}} P_\varphi(x_1)$, $P_\varphi(x_1)$ will have a non-zero value iff at both $x_1 = 0, P_\varphi(x_1) > 0$

and $x_1 = 1, P_\varphi(x_1) > 0$, in this case we will multiply both values and hence it is true iff all the values are greater than zero.

We suppose it for $n = m$. We suppose $Q_1x_1Q_2x_2....Q_mx_m\varphi(x_1, x_2, x_3, ..., x_m) \in TQBF$ if and only if $\sum_1 \Pi_2 \sum_3 ....\Pi_m P_\varphi(\vec{x}) \neq 0$, where $\sum_i$ represents $\sum_{x_i}$ and Π_i represents Π_{x_i}

Now, we have to show that it holds for $n = m + 1$.
$Q_1x_1Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(x_1, x_2, x_3, ..., x_m, x_{m+1}) \in TQBF$ then, it can be written in polynomial form. Without loss of generality, we can assume that $Q_1 = \exists$.

$\iff \big[Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(0, x_2, x_3, ..., x_m, x_{m+1})\big] \in TQBF$ OR

$\big[Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(1, x_2, x_3, ..., x_m, x_{m+1})\big] \in TQBF$

$\iff \big[\Pi_2 \sum_3 ....\Pi_{m+1} P_\varphi(0, x_2, x_3, ..., x_m, x_{m+1}) \neq 0\big]$ OR

$\big[\Pi_2 \sum_3 ....\Pi_m P_\varphi(1, x_2, x_3, ..., x_m, x_{m+1}) \neq 0\big]$

$\iff \big[\Pi_2 \sum_3 ....\Pi_{m+1} P_\varphi(0, x_2, x_3, ..., x_m, x_{m+1})\big] +$

$\big[\Pi_2 \sum_3 ....\Pi_m P_\varphi(1, x_2, x_3, ..., x_m, x_{m+1})\big] \neq 0$ (.. as each of the expression is a non-negative value).

$\sum_1 \Pi_2 \sum_3 ....\Pi_m P_\varphi(x_1, x_2, x_3, ..., x_m, x_{m+1}) \neq 0.$

Similarly, when $Q_1 = \Pi$ then

$Q_1x_1Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(x_1, x_2, x_3, ..., x_m, x_{m+1}) \in TQBF$ then, it can be written in polynomial form.

$\iff \big[Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(0, x_2, x_3, ..., x_m, x_{m+1})\big] \in TQBF$ AND

$\big[Q_2x_2....Q_mx_mQ_{m+1}x_{m+1}\varphi(1, x_2, x_3, ..., x_m, x_{m+1})\big] \in TQBF$

$\iff \big[\Pi_2 \sum_3 ....\Pi_{m+1} P_\varphi(0, x_2, x_3, ..., x_m, x_{m+1}) \neq 0\big]$ AND

$\big[\Pi_2 \sum_3 ....\Pi_m P_\varphi(1, x_2, x_3, ..., x_m, x_{m+1}) \neq 0\big]$

$\iff \big[\Pi_2 \sum_3 ....\Pi_{m+1} P_\varphi(0, x_2, x_3, ..., x_m, x_{m+1})\big] *$

$\big[\Pi_2 \sum_3 ....\Pi_m P_\varphi(1, x_2, x_3, ..., x_m, x_{m+1})\big] \neq 0$ (.. as each of the expression must be a non-negative value and non-zero).

$\Pi_1\Pi_2 \sum_3 ....\Pi_m P_\varphi(x_1, x_2, x_3, ..., x_m, x_{m+1}) \neq 0.$

Thus, we have shown the inductive step to be true.

Hence, we can see that $Q_1x_1Q_2x_2....Q_nx_n\varphi(x_1, x_2, x_3, ..., x_n) \in TQBF$ if and only if $\big(\sum_{x_1\in\{0,1\}} or \Pi_{x_1\in\{0,1\}}\big)\big(\sum_{x_2\in\{0,1\}} or \Pi_{x_2\in\{0,1\}}\big)....\big(\sum_{x_n\in\{0,1\}} or \Pi_{x_n\in\{0,1\}}\big)P_\varphi(x_1, x_2, ..., x_n) = k$, where k is some non-zero positive integer.

Recall $\#SAT \in IP$ argument ($LFKN'90$) using Sum-Check protocol : $\sum_{x_1\in\{0,1\}} ..... \sum_{x_n\in\{0,1\}} P_\varphi(x_1, x_2, x_3, ..., x_n) = k$.

Without loss of generality let the input instance be $\exists x_1 \forall x_2 \exists x_3 ....\forall x_n \varphi(x_1, x_2, x_3, ..., x_n)$, where φ is a 3-CNF with m clauses. Let us attempt to apply the Sum-Check protocol.

$$\sum_{x_1} \Pi_{x_2} \sum_{x_3} ....\Pi_{x_n} P_\varphi(x_1, x_2, x_3, ..., x_n) = k \qquad .... 1.$$

Prover is trying to convince the verifier that the above equality holds. Verifier should reject with high probability if the above equality does not hold.

In the beginning, both the verifier and the prover find out the arithmetic expression P_φ from φ (degree of $P_\varphi \leq 3m$). Then, the prover sends a k, and a large prime p to the verifier. Then, the rest of the computation happens over $\mathbb{F}_p$.

Let $h(x_1) = \Pi_{x_2} \sum_{x_3} ....\Pi_{x_n} P_\varphi(x_1, x_2, ..., x_n)$.

We cannot, straightaway, repeat the previous Sum-Check protocol because of the following issue :

Issue : $deg_{x_1}(h(x_1))$ can be exponentially large (may not be a polynomial) because of the products. Hence, it is not immediately clear if the prover can send h to the verifier.

$(Shamir'90)$ carried on from here and sometime later from $(LFKN'90)$ got the IP = PSPACE result by fixing this issue.

Definition : (Linearizing Operator) ($x_i \in \{0,1\}$) As we see that the degree of the polynomial can be exponential (because of the $\forall$ quantifier), hence we try to come up with a technique that controls the degree of this polynomial. We are just dealing with $x_i \in \{0,1\}$, hence we know no matter what the degree of x_i be in the polynomial, it can be replaced by just x_i because $x_i^k = x_i$ for $x_i \in \{0,1\}$. Hence, we define the linearizing operator as follows :

$L_{x_i} P(x_1, x_2, ..., x_n) = x_i P(x_1, x_2, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) + (1 - x_i) P(x_1, x_2, ..., x_{i-1}, 0, x_{i+1}, ..., x_n)$. L_{x_i} can be written as L_i for short. We adjust/modify the previous Sum-Check expression to the following expression(using linearizing operators to control the degree) :

$$\sum_{x_1} L_{x_1} \Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(x_1, x_2, x_3, ...x_n) = k. \qquad .... 2.$$

The verifier knows this LHS expression and asks the prover for its value. The prover returns some value, say k, and if the value is non-zero he asks the prover to send a polynomial $h'(x_1)$ that is univariate in x_1 :

$h'(x_1) := L_{x_1} \Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(x_1, x_2, x_3, ...x_n)$ is $h'(x_1)$ (we will see later).

Now, the prover and the verifier have to check this formula instead of the earlier one.

Observation : (1) is true iff (2) is true for $\{0,1\}$ values of x_i.

e.g. Verifier asks the prover to send $h'(x_1) = L_{x_1} \Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(x_1, x_2, x_3, ...x_n)$ ($deg_{x_1}(h'(x_1)) \leq 1$).

Now, the prover sends some $s(x_1)$ (an honest prover sends $s(x_1) = h'(x_1)$). Verifier checks if $s(0) + s(1) = k$, if not he rejects. Otherwise, he picks a random $a \in \mathbb{F}_p$ and asks the prover to prove :

$$\left[L_{x_1} \Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(x_1, x_2, x_3, ...x_n)\right]_{x_1 = a} = s(a).$$

Verifier asks for the polynomial $h''(x_1) (deg(h''(x_1)) \leq 2)$, where

$h''(x_1) = \Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(\vec{x})$.

Prover returns some $s'(x_1)$. Once again, an honest prover sends $s'(x_1) = h''(x_1)$.

Verifier checks if $\left[L_{x_1} s'(x_1)\right]_{x_1 = a} = s(a)$. Suppose $s'(x_1) = h''(x_1)$. Then, $L_{x_1} s'(x_1) = h'(x_1)$, and verifier would be checking if $\left[L_{x_1} s'(x_1)\right]_{x_1 = a} = h'(a) = s(a)$ for a random $a \in \mathbb{F}_p$. However, $h'(a) \neq s(a)$ with high probability unless $h'(x_1) = s(x_1)$. Hence, if $s'(x_1) = h''(x_1)$, then a dishonest prover can try to not get caught by sending $h'(x_1) = s(x_1)$, but in this scenario it would have already gotten caught at the prior check $s(0) + s(1) = k$. Therefore, a dishonest prover's only chance is to send a $s'(x_1)$ different from $h''(x_1)$.

Verifier picks a random $a' \in_R \mathbb{F}_p$. Then, he asks the prover to prove:

$$\left[\Pi_{x_2} L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(\vec{x})\right]_{x_1 = a'} = s'(a').$$

i.e. $\Pi_{x_2} \left[L_{x_1} L_{x_2} \sum_{x_3} L_{x_1} L_{x_2} L_{x_3} .....\Pi_{x_n} L_{x_1} L_{x_2} L_{x_3} ...L_{x_n} P_\varphi(\vec{x})\right]_{x_1 = a'} = s'(a')$.

Verifier asks for the polynomial : $\underbrace{\big[L_{x_1}L_{x_2}\sum_{x_3}L_{x_1}L_{x_2}L_{x_3}.....\Pi_{x_n}L_{x_1}L_{x_2}L_{x_3}...L_{x_n}P_\varphi(\vec{x})\big]_{x_1=a'}}_{g(x_2)}$.

It is denoted as $g(x_2); deg_{x_2}g(x_2) \leq 1$.

Prover sends some $r(x_2)$ to the verifier.

Verifier checks if $r(0).r(1) = s'(a')$. If not, he rejects otherwise he picks a random $a_2 \in_R \mathbb{F}_p$ and asks the prover to prove the following :

$\big[L_{x_1}L_{x_2}\sum_{x_3}L_{x_1}L_{x_2}L_{x_3}.....\Pi_{x_n}L_{x_1}L_{x_2}L_{x_3}...L_{x_n}P_\varphi(\vec{x})\big]_{x_1=a',x_2=a_2} = r(a_2)$.

i.e. $\big[L_{x_1}\big\{\big[L_{x_2}\sum_{x_3}L_{x_1}L_{x_2}L_{x_3}.....\Pi_{x_n}L_{x_1}L_{x_2}L_{x_3}...L_{x_n}P_\varphi(\vec{x})\big]_{x_2=a_2}\big\}\big]_{x_1=a'} = r(a_2)$.

The last step can be explained as follows :

Consider $\big[L_{x_1}L_{x_2}\sum_{x_3}L_{x_1}L_{x_2}L_{x_3}.....\Pi_{x_n}L_{x_1}L_{x_2}L_{x_3}...L_{x_n}P_\varphi(\vec{x})\big]_{x_1=a',x_2=a_2}$ to be

$\big[L_{x_1}\underbrace{L_{x_2}M(x_1,x_2)}_{M'(x_1,x_2)}\big]_{x_1=a',x_2=a_2}$ (and in this consider $L_{x_2}M(x_1,x_2)$ as $M'(x_1,x_2)$).

Applying linearizing operator L_{x_1}, we get :

= $\big[x_1M'(1,x_2)+(1-x_1)M'(0,x_2)\big]_{x_1=a',x_2=a_2}$

= $\big[x_1M'(1,a_2)+(1-x_1)M'(0,a_2)\big]_{x_1=a'}$

= $\big[L_{x_1}\big[M'(x_1,x_2)\big]_{x_2=a_2}\big]_{x_1=a'}$

= $\big[L_{x_1}\big[L_{x_2}M(x_1,x_2)\big]_{x_2=a_2}\big]_{x_1=a'}$

Now, Verifier asks for $\big\{\big[\underbrace{L_{x_2}\Pi_{x_3}.....\Pi_{x_n}L_{x_1}L_{x_2}L_{x_3}...L_{x_n}P_\varphi(\vec{x})\big]_{x_2=a_2}}_{h'''(x_1)}\big\}$. We call it $h'''(x_1)$.

Prover sends some $s''(x_1)$ to the verifier.

Verifier checks if $\big[L_{x_1}s''(x_1)\big]_{x_1=a'} = r(a_2)$.

Suppose $s''(x_1) = h'''(x_1)$. Then,

$\Rightarrow \big[L_{x_1}h'''(x_1)\big]_{x_1=a'} = g(a_2) = r(a_2)$. However, $g(a_2) \neq r(a_2)$ with high probability as a dishonest prover will send $r(x_2) \neq g(x_2)$ or else it'll get caught at the check $r(0).r(1) = s'(a')$. Hence, a dishonest prover sends $s''(x_1) \neq h'''(x_1)$.

Verifier picks a random $b_1 \in_R \mathbb{F}_p$ and checks if $h'''(b_1) = s''(b_1)$

i.e. $\big[L_{x_2}\Pi_{x_3}....\Pi_{x_n}L_{x_1}L_{x_2}.....L_{x_n}P_\varphi(\vec{x})\big]_{x_2=a_2,x_1=b1} = s''(b_1)$.

And, in this way, the process continues and one by one each operator is shaved off. We will make illustrative argument more formal and precise in the next lecture.

24.2 References

[1] S. ARORA and B. BARAK "Computational Complexity: A Mordern Approach," *Cambridge University*

Press, 2009

[2] http://crypto.cs.mcgill.ca/ crepeau/COMP647/2007/TOPIC01/Shamir-IP=PSPACE.pdf