Lecture 19

In this lecture, we will prove that **PSPACE** $\subseteq$ **IP**. We will essentially use the same algebraic ideas from the last lecture where we proved $\sharp$**P** $\subseteq$ **IP**.

1 PSPACE-Complete Language: TQBF

For a 3-CNF boolean formula $\phi(x_1, x_2, \ldots, x_n)$, we may think of its satisfiability problem as determining the truth value of the statement

$$\exists x_1 \exists x_2, \ldots, \exists x_n \quad \phi(x_1, x_2, \ldots, x_n).$$

We can generalize this idea to allow universal quantifiers in additional to existential quantifiers to get formulas denoted as quantified boolean formulas. For example, $\forall x_1 \exists x_2 \quad (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ is a quantified boolean formula. In fact, it's easy to see that it's a true quantified boolean formula. Now consider the language of all true quantified boolean formulas:

$$\text{TQBF} = \{\Phi : \quad \Phi \text{ is a true quantified boolean formula}\}.$$

It is known that TQBF is **PSPACE**-complete. Thus, if we have an interactive-proof protocol for recognizing TQBF, then we have a protocol for recognizing any language in **PSPACE**. In the rest of the lecture, we will provide a protocol for recognizing TQBF.

2 Arithmetization of TQBF

We will consider the following quantified boolean formula:

$$\Phi = \forall x_1 \exists x_2 \forall x_3 \ldots \forall x_n \quad \phi(x_1, x_2, \ldots, x_n),$$

where $\phi(x_1, x_2, \ldots, x_n)$ is a $3 - CNF$ boolean formula. Without loss of generality, we can assume that all the quantified boolean formulas given to us have this form. The main idea in the interactive protocol for recognizing TQBF is the same as proving $\sharp$**P** $\subseteq$ **IP** from the last lecture: we will first arithmetize the quantified boolean formula and then the prover will convince the verifier that the arithmetized expression evaluates to 1. In the following, all the random elements are drawn from a field $\{0, 1, \ldots, p-1\}$, where p is sufficiently large.

2.1 Naive Solution

So how do we arithmetize the quantified formula? we will first show a naive solution and point out its problems. To begin with, we know from the last lecture how to arithmetize $\phi(x_1, x_2, \ldots, x_n)$ to obtain the polynomial $F(x_1, x_2, \ldots, x_n)$. Recall that F has degree $3m$ in each variable (m is the number of clauses in ϕ) and agrees with ϕ for all $0-1$ assignments to the n variables. Now we read the quantifiers from right to left and consider the expression

$\forall x_n \phi(x_1, x_2, \ldots, x_n)$. This expression has $n-1$ free variables, and for each substitution of values to these variables the expression itself either evaluates to true or false. We're looking for a polynomial that has the same behavior. Using F we can now write a new polynomial $P_{\forall x_n} F(x_1, x_2, \ldots, x_n)$, which is equal to

$$F(x_1, x_2, \ldots, x_{n-1}, 0) \cdot F(x_1, x_2, \ldots, x_{n-1}, 1).$$

Next consider the previous quantifier $\exists x_{n-1}$. We want to find a polynomial representation for $\exists x_{n-1} \forall x_n \phi(x_1, x_2, \ldots, x_n)$, given the polynomial representation $G(x_1, x_2, \ldots, x_{n-1})$ for $\forall x_n \phi(x_1, x_2, \ldots, x_n)$. The following polynomial, which we denote by $P_{\exists x_{n-1}} G(x_1, x_2, \ldots, x_{n-1})$, satisfies the condition:

$$1 - (1 - G(x_1, x_2, \ldots, x_{n-2}, 0)) \cdot (1 - G(x_1, x_2, \ldots, x_{n-2}, 1)).$$

In the notations introduced above, the polynomial for $\exists x_{n-1} \forall x_n \phi(x_1, x_2, \ldots, x_n)$ is

$$P_{\exists x_{n-1}} P_{\forall x_n} F(x_1, x_2, \ldots, x_n).$$

In general, we transform the quantified boolean formula Φ into an arithmetic expression using operators $P_{\forall u}$ and $P_{\exists v}$ as follows:

- Turn the 3-CNF formula $\phi(x_1, x_2, \ldots, x_n)$ into the polynomial $F(x_1, x_2, \ldots, x_n)$ as done in the last lecture.
- Replace the quantifier $\exists x_i$ by the operator $P_{\exists x_i}$.
- Replace the quantifier $\forall x_i$ by the operator $P_{\forall x_i}$.

The final expression after the transformation always evaluates to 0 or 1. It evaluates to 1 if and only if the quantified boolean formula Φ is true.

We have thus arrived at a way of representing quantified boolean formulas using arithmetic operations. Now consider the arithmetic expression after the transformation

$$P_{\forall x_1} P_{\exists x_2} \ldots P_{\forall x_n} F(x_1, x_2, \ldots, x_n)$$

and it evaluates to either 0 or 1. Let us apply the idea from the last lecture to derive a protocol and see what happens. In the first step, the prover eliminates the first operator $P_{\forall x_1}$ and sends the verifier the polynomial $G_1(x_1)$ corresponding to the rest of the arithmetic expression:

$$G_1(x_1) = P_{\exists x_2} P_{\forall x_3} \ldots P_{\forall x_n} F(x_1, x_2, \ldots, x_n).$$

Upon receiving the polynomial $\hat{G}_1(x_1)$ (which the prover claims to be $G_1(x_1)$), the verifier checks that $\hat{G}_1(0) \cdot \hat{G}_1(1) = 1$. The verifier then picks a random value r_1, computes $\beta_1 = \hat{G}_1(r_1)$ and sends r_1 to the prover. The prover then needs to convince the verifier that

$$\beta_1 = P_{\exists x_2} P_{\forall x_3} \ldots P_{\forall x_n} F(r_1, x_2, \ldots, x_n).$$

The prover then proceeds by eliminating the operator one after another, and in the end the verifier checks that $\beta_n = F(r_1, r_2, \ldots, r_n)$.

However, since each operator potentially doubles the degree of each variable in the expression, the degree of the polynomial $G_1(x_1)$ in the end can very well be exponential. This means that the prover will have to send the verifier exponentially many coefficients, but the polynomially bounded verifier will not be able to read them all.

2.2 Revised Solution

We will ensure that the degree of any variable in any intermediate stage of the transformation never goes above two. The arithmetization given above clearly does not satisfy this property. Yet there is a simple way to fix this. At any stage of the transformation, we have a polynomial $J(x_1, x_2, \ldots, x_n)$ where some variables' degree might exceed two. Normally, we can't expect to transform J into a new polynomial $J'(x_1, x_2, \ldots, x_n)$ where the degree of any variable is at most two while still evaluating to the same value at all points. Yet, notice here that we only need J' to agree with J on all $0-1$ assignments. What J' does at any other point is of no interests to us. The key observation then is that for $x = 0$ or $x = 1$, $x^k = x$ for all positive integers k. The desired polynomial J' can thus be obtained from J by erasing all exponents, that is, replacing them with 1. For example, if $J(x_1, x_2, x_3) = x_1^3 x_2^4 + 5x_1 x_2^3 + x_2^6 x_3^2$, the transformed polynomial J' is $6x_1 x_2 + x_2 x_3$. In general, we define a new operator, Rx_i, which when applied to a polynomial reduces the exponents of x_i to 1 at all occurrences. More formally, we have

$$
\begin{aligned}
Rx_i J(x_1, \ldots, x_n) = x_i \cdot J(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + \\
(1 - x_i) \cdot J(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).
\end{aligned}
$$

In this notation, we have

$$
J'(x_1, \ldots, x_n) = Rx_1 Rx_2 \ldots Rx_n J(x_1, \ldots, x_n).
$$

In the revised arithmetization process, the only change we make is to apply the reduce operators whenever the degree of any variable can potentially be above 1 in the polynomial obtained from the previous step. In particular, we apply the reduce operators after the 3-CNF formula ϕ is arithmetized into the polynomial $F(x_1, \ldots, x_n)$. Also, since the other two operators, $P_{\forall u}$ and $P_{\forall v}$, might double the degree of some variables in the transformation, we apply the reduce operators after each application of $P_{\forall u}$ or $P_{\exists v}$. This revision will ensure that the degree of any variable at any intermediate stage never goes above 2. The degree of some variable might reach 2 after a $P_{\forall u}$ or $P_{\exists v}$ operator, but will be reduced back to 1 after the reduce operators are applied.

3 The Interactive Protocol

Using the revised arithmetization, we arithmetize the quantified boolean formula of the form

$$
\Phi = \forall x_1 \exists x_2 \forall x_3 \ldots \forall x_n \phi(x_1, \ldots, x_n)
$$

into

$$
E = P_{\forall x_1} Rx_1 P_{\exists x_2} Rx_1 Rx_2 P_{\forall x_3} Rx_1 Rx_2 Rx_3 P_{\exists x_4} \ldots P_{\forall x_n} Rx_1 \ldots Rx_n F(x_1, \ldots, x_n).
$$

Now the idea of the interactive protocol from the last lecture can be put to use without any trouble. The prover and the verifier communicate to eliminate one operator at each stage. In the end, no operators are left and the verifier checks the required condition.

Initially, the prover eliminates the first operator $P_{\forall x_1}$ and sends the verifier the polynomial $G_1(x_1)$ corresponding to the rest of the arithmetic expression:

$$G_1(x_1) = Rx_1 P_{\exists x_2} \ldots Rx_n F(x_1, \ldots, x_n).$$

Notice that this is a polynomial of degree 1 instead of exponential degree in the original naive arithmetization. Upon receiving the polynomial $\hat{G}_1(x_1)$(which the prover claims to be $G_1(x_1)$), the verifier checks that $P_{\forall x_1}\hat{G}_1(x_1) = \hat{G}_1(0) \cdot \hat{G}_1(1) = 1$. The verifier then picks a random value r_1, computes $\beta_1 = \hat{G}_1(r_1)$ and sends r_1 to the prover. (In this stage, the verifier actually doesn't have to send r_1 to the prover, but for consistency with later rounds we let the verifier send r_1 to the prover. The soundness of the protocol remains.) The prover then needs to convince the verifier that

$$\beta_1 = Rx_1 P_{\exists x_2} \ldots Rx_n F(r_1, x_2, \ldots, x_n).$$

The prover then eliminates the second operator Rx_1 and sends the verifier the polynomial $G_2(x_1)$ corresponding to the rest of the arithmetic expression:

$$G_2(x_1) = P_{\exists x_2} \ldots Rx_n F(x_1, \ldots, x_n).$$

Notice that this is a polynomial of degree at most 2. Upon receiving the polynomial $\hat{G}_2(x_1)$(which the prover claims to be $G_2(x_1)$), the verifier checks that $(Rx_1 \hat{G}_2(x_1))[r_1] = \hat{G}_1(r_1)$, where $[r_1]$ means that the polynomial in the parenthesis with free variable x_1 is evaluated at point r_1. The verifier then picks a random value r_2 for x_1, computes $\beta_2 = \hat{G}_1(r_2)$ and sends r_2 to the prover.

In general, when the verifier needs to be convinced that

$$\beta = (Rx_i P(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, x_i))[\alpha_i],$$

he asks for the polynomial $G(x_i) = P(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, x_i)$, and gets $\hat{G}(x_i)$. ($\alpha_j (1 \leq j \leq i)$ is the most recent random value the verifier assigned to variable x_j.) He verifies that $(Rx_i \hat{G}(x_i))[\alpha_i] = \beta$ and computes $\beta' = \hat{G}(\alpha_i')$ by choosing a new random value α_i' for x_i. Now the verification is reduced to whether

$$\beta' = P(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_i').$$

The case of the $P_{\forall u}$ or $P_{\exists v}$ operator is the same as in the last lecture where we proved $\sharp\mathbf{P} \subseteq \mathbf{IP}$.

The prover proceeds by eliminating the operators one after another, and in the end the verifier checks that $q(\alpha_n) = F(\alpha_1, \ldots, \alpha_n)$, where q is the polynomial the prover sends to the verifier in the last round and α_i is the most recent random value the verifier assigned to variable x_i.

4 Analysis

THEOREM 1
(a) If $\Phi \in TQBF$, then there exists a prover such that $Pr[Verifier\ accepts] = 1$.
(b) If $\Phi \notin TQBF$, then for all provers $Pr[Verifier\ accepts] \leq 1/2$.

PROOF: Part (a) is obvious. For part (b), the analysis is the same as in the case of $\sharp\mathbf{P} \subseteq \mathbf{IP}$ from the last lecture. The error can be introduced only when the verifier happens to pick a root of some low degree polynomial at some round. While dealing with the last n reduce operators, the polynomials may have degree at most $3m$. For the remaining $n(n-1)/2$ reduce operators, the degree is at most 2. For every other operator (n operators of the form $P_{\forall u}$ or $P_{\exists v}$) the degree of the polynomial is 1. Thus, the sum of the degrees of all the polynomials encountered in the interaction is at most $3mn + n^2$. Thus, if we choose the size of our field p large enough, the probability of error is at most $(3mn + n^2)/p \leq 1/2$. $\Box$