Notes for Lecture 15

In this lecture we will see that **IP** = **PSPACE**. We start by giving an interactive proof system for UNSAT, and therefore show that **coNP** $\subseteq$ **IP**. In fact, as we will see, the same proof system with minor modifications can be used for #SAT. Since #SAT is #**P**-complete, and since **PH** $\subseteq$ $\mathbf{P}^{\#\mathbf{P}}$, then **PH** $\subseteq$ **IP**. Finally, we generalize this protocol in order to solve QSAT, and thus show that **PSPACE** $\subseteq$ **IP**.

1 UNSAT $\subseteq$ IP

Let ϕ be a formula with m clauses over the variables $x_1, \ldots, x_n$. Let $N > 2^n \cdot 3^m$ be a prime number. We translate ϕ to a polynomial p over the field $\pmod N$ in the following way. Literals are translated as follows: $x_i \to x_i$, $\overline{x_i} \to (1 - x_i)$. A clause is translated to the sum of the (at most 3) expressions corresponding to the literals of the clause. Finally, p is taken as the product of the m expressions corresponding to the m clauses. Notice that each literal is of degree 1, and therefore p is of degree at most m. Furthermore, for a 0-1 assignment, p evaluates to 0 if this assignment does not satisfy ϕ, and to a non-zero number if this assignment does satisfy ϕ where this number can be at most 3^m. Hence, ϕ is unsatisfiable iff

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(x_1, \ldots, x_n) \equiv 0 \pmod N$$

Also, the translation to p can be done in polynomial time, and if we take $N = 2^{O(n+m)}$ then elements in the field can be represented by $O(n+m)$ bits, and thus one evaluation of p in any point in $\{0, \ldots, N-1\}^m$ can be computed in polynomial time.

We now describe a protocol that enables the prover to convince the verifier that ϕ is unsatisfiable (see also Figure 1 for a sketch of the protocol). At the first round, the prover sends N, a proof that it is prime, and an m-degree univariate polynomial $q_1(x) = \sum_{x_2,\ldots,x_n \in \{0,1\}} p(x, x_2, \ldots, x_n)$. At this point, the verifier verifies the primality proof as well as checking that $q_1(0) + q_1(1) = 0$. If any of the checks fail then the verifier rejects. Otherwise, it continues by choosing a random $r_1 \in \{0, \ldots, N-1\}$ and sending it to the prover. The prover responds by sending an m-degree univariate polynomial $q_2(x) = \sum_{x_3,\ldots,x_n \in \{0,1\}} p(r_1, x, x_3, \ldots, x_n)$. Now, the verifier checks if $q_2(0) + q_2(1) = q_1(r_1)$. If not, it rejects, and otherwise it continues by choosing a random $r_2 \in \{0, \ldots, N-1\}$ and sending it to the prover. The protocol continues in this way where the ith message that the prover sends is a polynomial $q_i(x) = \sum_{x_{i+1},\ldots,x_n \in \{0,1\}} p(r_1, \ldots, r_{i-1}, x, x_{i+1}, \ldots, x_n)$. The verifier responds by checking that $q_i(0) + q_i(1) = q_{i-1}(r_{i-1})$, rejecting if it is not, and otherwise, choosing a random $r_i \in \{0, \ldots, N-1\}$ and sending it to the prover. At the last round, the prover sends $q_n(x) = p(r_1, \ldots, r_{n-1}, x)$. The verifier checks that $q_n(0) + q_n(1) = q_{n-1}(r_{n-1})$, chooses a random $r_n \in \{0, \ldots, N-1\}$, and checks that $q_n(r_n) = p(r_1, \ldots, r_n)$. The verifier rejects if either of the checks fail, and accepts otherwise. The verifier can indeed perform the last check since as mentioned before, it can

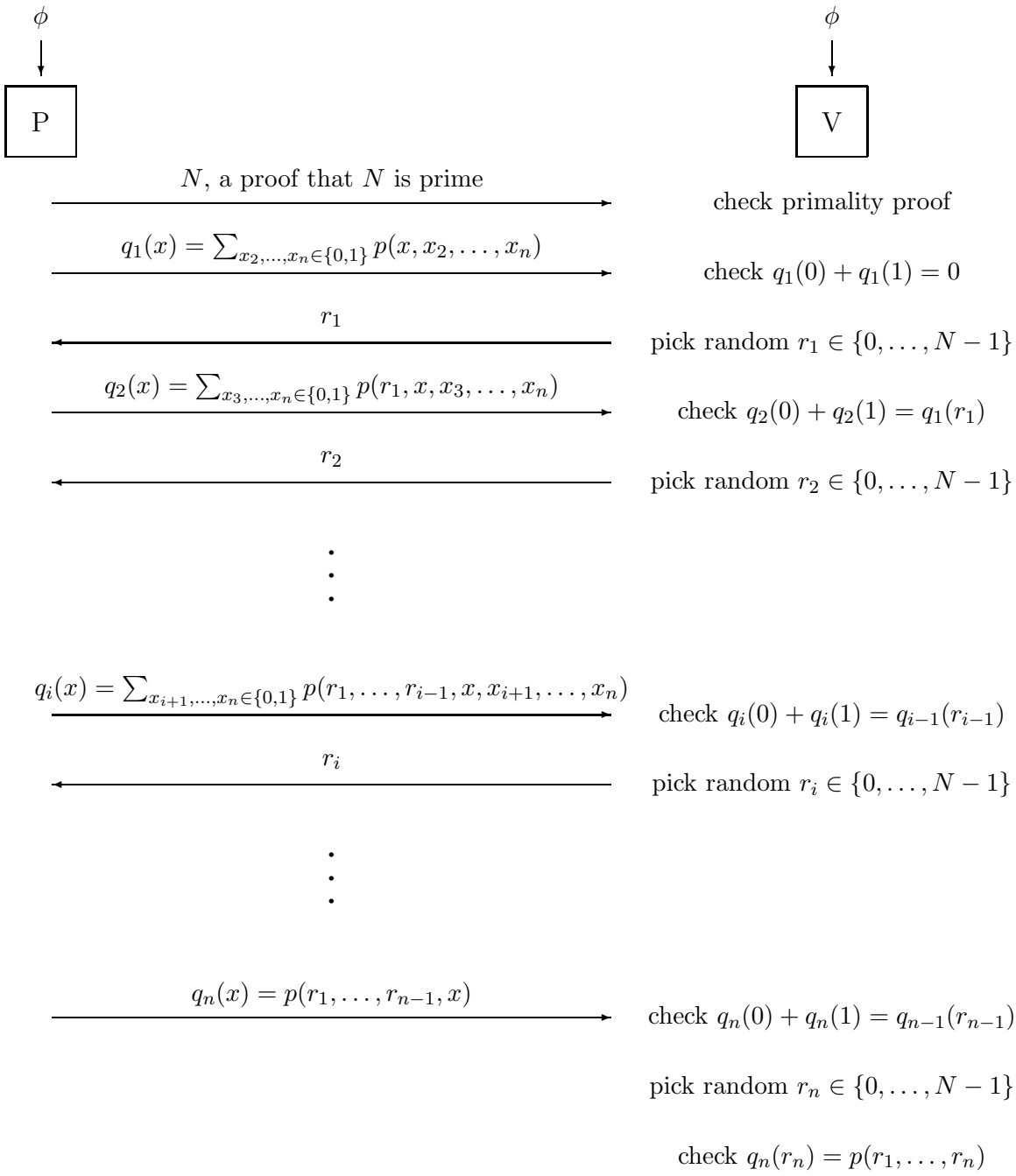

Figure 1: The protocol for proving unsatisfiability

translate ϕ to p and evaluate it at the point $(r_1, \ldots, r_n)$ in polynomial time. Also, the messages are of polynomial length because elements of the field can be written with $O(m+n)$ bits, and since each polynomial has m coefficients.

We now show the correctness of this proof system. If ϕ is unsatisfiable then the prover can make the verifier accept w.p. 1 by just following the protocol since $q_1(0) + q_1(1) = \sum_{x_1,\ldots,x_n \in \{0,1\}} p(x_1, \ldots, x_n) \equiv 0 \pmod{N}$. It is clear that the rest of the checks will succeed if the prover send the q_i's accordingly. We have to show that if ϕ is satisfiable then no matter what polynomials the prover sends, the verifier will reject with high probability. We write $p_i(x)$ for the polynomial that a prover who follows the protocol sends in round i. Since ϕ is satisfiable then $\sum_{x_1,\ldots,x_n \in \{0,1\}} p(x_1, \ldots, x_n) \not\equiv 0 \pmod{N}$. Therefore, $p_1(0) + p_1(1) \neq 0$, and hence, if the prover sends $q_1 = p_1$ then the verifier will reject. If the prover sends $q_1 \neq p_1$ then since both are degree-m polynomials then they agree on at most m places. Thus, there is probability $\geq 1 - \frac{m}{N}$ that $p_1(r_1) \neq q_1(r_1)$. Suppose that indeed $p_1(r_1) \neq q_1(r_1)$. If the prover then sends $q_2 = p_2$ then the verifier will reject because $q_2(0)+q_2(1) = p_1(r_1) \neq q_1(r_1)$. Thus, the prover must send $q_2 \neq p_2$. Again, in that case, q_2 and p_2 will differ on a fraction $\geq 1 - \frac{m}{N}$ of the elements of $\{0, \ldots, N-1\}$, so $p_2(r_2) \neq q_2(r_2)$ with probability $\geq 1 - \frac{m}{N}$. We continue the argument in a similar way. If $q_i \neq p_i$ then w.p. $\geq 1 - \frac{m}{N}$, r_i is such that $p_i(r_i) \neq q_i(r_i)$. If so, then the prover must send $q_{i+1} \neq p_{i+1}$ in order for the verifier not to reject. At the end, if the verifier has not rejected before the last check, then w.p. $\geq 1 - (n-1)\frac{m}{N}$, $q_n \neq p_n$. If so, then w.p. $\geq 1 - \frac{m}{N}$ the verifier will reject since, again, $q_n(x)$ and $p(r_1, \ldots, r_{n-1}, x)$ differ on at least that fraction of the points. Thus, the total probability that the verifier will accept is at most $\frac{nm}{N}$.

2 A Proof System for #SAT

We now show that with a few minor modifications, the previous proof system can be used to prove that a formula has a given number of satisfying assignments. Suppose ϕ has k satisfying assignments. We wish to give a protocol s.t. if the prover gives k as an answer, he is able to continue with the protocol s.t. the verifier will accept w.p. 1, but if it gives another number as an answer, then no matter what messages the prover sends afterwards, the verifier will reject with high probability.

We first change the way we translate ϕ to the polynomial p. The change is only in the way we translate clauses. Instead of translating a clause to the sum of the literals, we translate the clause $(z_1 \vee z_2 \vee z_3)$ to $1-(1-z_1)(1-z_2)(1-z_3)$. Notice that if a 0-1 assignment to the variables satisfies the clause, then the value of this expression will be 1, and if the assignment does not satisfy the clause, this value will be 0. Hence, 0-1 assignments that satisfy ϕ will be evaluated to 1 by p, while as before, the 0-1 assignments that do not satisfy ϕ will be evaluated to 0. Notice that now the degree of p is $3m$ instead of m, but now the sum over all evaluations of 0-1 assignments is equal to the number of satisfying assignments of ϕ. For the same reason, it is now enough to take $N > 2^n$.

In the current protocol, the prover starts by sending k, and then continues as in the previous protocol (with the updated definition of p). After receiving the first message, the verifier checks if $q_1(0) + q_1(1) = k$ instead of checking $q_1(0) + q_1(1) = 0$, and follows the rest of the previous protocol. If k is indeed the correct number of satisfying assignments, then the prover can continue by sending $q_i = p_i$ at each round, and the verifier will accept

with probability 1. If k is not the correct number of satisfying assignments, then according to the same analysis as in the previous section, the verifier will accept with probability at most $\frac{3mn}{N}$.

3 A Proof System for QBF

3.1 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.

3.2 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 TQBFis the same as proving $\sharp\mathbf{P} \subseteq \mathbf{IP}$ 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.

3.2.1 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.

3.2.2 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 $6x_1x_2 + x_2x_3$. 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.

3.3 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 $\hat{G}(x_i)$. ($\alpha_j (1 \le j \le i)$ 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.

3.4 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 $Pr[Verifier\ accepts] \le 1/2$.

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 $(3mn + n^2)/p \leq 1/2$. □

Exercises

1. Show that **IP** $\subseteq$ **PSPACE**.

   [Hint: it is easier to show the following stronger claim: that given a verifier V and an input x, we can compute the maximum, over all provers' strategies P^*, of the probability that V accepts x while interacting with P^*.]