IP=PSPACE

Aditya Kumar, Shreyas Gupta

CSA, IISc, Bangalore

December 25, 2020

Outline

Proof in mathematics

We know a standard notion of a proof, where to prove a statement,

- a Prover writes a sequence of statements, known as proof.
- A Verifier can check the correctness of proof in following way
  1. there are finitely many statements in the proof
  2. every statement in the proof is either an axiom or can be logically deduced from previous statements.
  3. the last statement in proof is the statement we wanted to prove
- The statement is true if there exists a proof, for which verifier returns accept

Interactive Proofs

This usual notion proof can be generalized in the following sense:

- This usual "non-interactive" framework can be seen as an interaction between the "Prover(P)" and the "Verifier(V)" where P sends a message to V, V performs some computation on the message, and finally returns true or false
- This notion can be naturally generalized into an interactive framework, where the proof can be considered as many rounds of interaction between P and V

Connection with computational complexity

In complexity theory, we are interested in statements of the form "$S_L(x) := x \in L$". Following points can be noticed:

- $L \in \mathcal{NP}$ if there exists a proof of $S_L(x)$ of length $|x|^d$ for some $d \in \mathbb{N}$ iff $x \in L$
- If we look at usual non-interactive proofs and make the verifier randomized, then the class of languages it defines coincides with BPP.
- Somehow, a combination of both randomization and multi-round interaction will make the class IP interesting.

To formalize interactive proofs, we need to model the P and V

Interaction of deterministic functions

Let $P, V : \{0,1\}^* \to \{0,1\}^*$ be functions. A k-round interaction of P, V, on input $x \in \{0,1\}^*$. denoted by $< P, V > (x)$, is sequence of following strings $a_1, a_2, \ldots, a_k \in \{0,1\}^*$ defined as follows:

$$
\begin{aligned}
a_1 &= V(x) \\
a_2 &= P(x, a_1) \\
&\vdots \\
a_{2i+1} &= V(x, a_1, \ldots, a_{2i}) \\
a_{2i+2} &= P(x, a_1, \ldots, a_{2i+1})
\end{aligned}
$$

The output of V (or P) at the end of the interaction denoted by $\text{out}_V < P, V > (x)$ is defined to be $V(x, a_1, a_2, \ldots, a_k)$

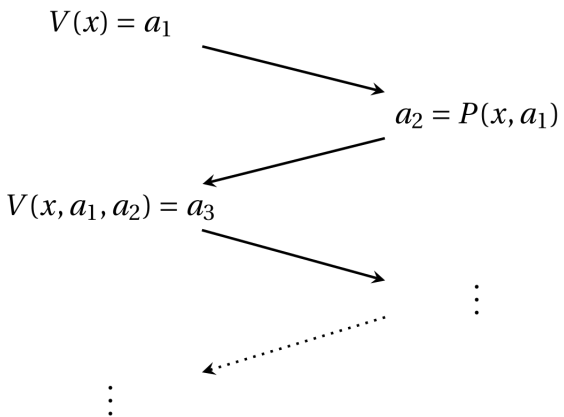

Figure 1: A general proof protocol

Deterministic proof system

We say that a k-round deterministic interactive proof system, if there is a deterministic TM V, that on input $x, a_1, a_2, \ldots, a_k$ runs in time polynomial in $|x|$, satisfying:

- (Completeness) $x \in L \implies \exists P : \{0,1\}^* \to \{0,1\}^*$ such that $out_V < V, P > (x) = 1$
- (Soundness) $x \notin L \implies \forall P : \{0,1\}^* \to \{0,1\}^*$ such that $out_V < V, P > (x) = 0$

We define class **dIP**, that contains all the languages with a $k(n)$-round deterministic interactive proof system, where k is a polynomial.
It turns out dIP=NP

The class IP

- We saw that increasing the number of rounds of interaction from 1 to polynomially many, didnot increase the power of our proof system.
- To realize full potential interaction, we need to let verifier be probabilistic.

Definition

Let $k : \mathbb{N} \to \mathbb{N}$ be some function with $k(n)$ computable in poly(n) time. A language L is in IP[k], if there is a Turing machine V, such that on inputs $x, a_1, a_2, \ldots, a_i,$, V runs in polynomial time in $|x|$ and such that

- (Completeness) $x \in L \implies \exists P \ Pr[out_V < V, P > (x) = 1] \geq 2/3$
- (soundness) $x \notin L \implies \forall P \ Pr[out_V < V, P > (x) = 1] \leq 1/3$

We define $IP = \cup_{c \geq 1} IP[n^c]$

Why is it interesting

- Whenever we define a class, we always want to see where it lies in our complexity hierarchy.
- But this problem is more interesting than this. The class IP and PSPACE are non relativizing.
- In 1988, Fortnow and Sipser proved that there exist oracles A and B such that $IP^A = PSPACE^A$ and $IP^B \neq PSPACE^B$
- Another consequence of this is that Co-IP=IP. This is not obvious because definition of IP is rather asymmetric.
- another interesting thing is that $Co - NP \subseteq IP$. Again, this is not obvious because no short certificate for x exists, if x is an element of Co-NP language. Therefore, making a prover in this case is not obvious.

Lemma

Let the language L be accpeted by the system (P,V), where V runs in time n^k for some $k \in \mathbb{N}$. If P is PSPACE computable, then L is in PSPACE.

Algorithm 1: Decide membership of L, assuming P is PSPACE-computable

Input: $w \in \{0,1\}^*$
$N \leftarrow |w|^k$
$c \leftarrow 0$
forall $r \in \{0,1\}^N$ **do**
    Simulate V using r as source of random bits, answering requests with P
    if *V accepts* **then**
        $c \leftarrow c + 1$
if $c \geq \frac{2}{3} \times 2^N$ **then**
    accept;
else
    reject;

- What if P is not PSPACE-computable? What if it is not even computable?

We will now show that P can always be modelled to be PSPACE-computable. It is important to note here what P exactly tries to achieve in this interactive proof system. It can be seen as a game between P and V, where goal of P is to maximize the probability of V accepting each input string $w \in L$.

Optimal prover

Definition

Let (P,V) be an interactive proof system, and let $w, w^{(i)}, v^{(i)} \in \{0,1\}^*$ be arbitrary. We denote by $\mathbb{P}_V(w^{(m+1)}|w, (w^{(i)})_{i=1}^m, (v^{(i)})_{i=1}^m)$ the conditional probability that the $(m+1)^{th}$ request of V has value $w^{(m+1)}$, given that the input is w and the first m requests have the values $w^{(i)}$ and answers $v^{(i)}$. We also denote by $\mathbb{P}_{(P,V)}(acc|w, (w^{(i)})_{i=1}^m, (v^{(i)})_{i=1}^m)$, the probability that V eventually accepts using the prover P, given same condiitons as above

Definition

An optimal prover for the verifier V is a prover function P_{opt} such that for all $w \in \{0,1\}^*$, all message histories $(w^{(i)})_{i=1}^m, (v^{(i)})_{i=1}^m)$ and all provers P, we have

$$\mathbb{P}_{(P_{opt},V)}(acc|w, (w^{(i)})_{i=1}^m, (v^{(i)})_{i=1}^m) \geq \mathbb{P}_{(P,V)}(acc|w, (w^{(i)})_{i=1}^m, (v^{(i)})_{i=1}^m)$$

Lemma

If the language L is accepted by the interactive proof system (P,V) and P_{opt} is an optimal prover for V, then L is accepted by the system (P_{opt}, V)

Lemma

Suppose that a verifier V runs in time n^k for some $k \in \mathbb{N}$. Then there exists a PSPACE-computable optimal prover for V.

Structure of proof for previous lemma:

- Given an input string and a request history, the prover P_{opt} first recalls its previous answers.
- It cycles through the answers it may possibly give
- for each of them, it calculates the probability that V eventually accepts if that answer is given (assuming that P behaves optimally from that point on).
- This calculation involves simulating V with all possible random choices it might make.
- P chooses the answer that maximizes the probability of acceptance.

Formalizing the proof

We will makes the following assumptions about V:

- All the requests of V, and the answers it accepts, are of length one.
- The number of requests that V sends during its computation on an input word w is a polynomial function of $|w|$ and we denote it by $M(|w|)$
- the last message of V is 1 if and only if it accepts the input.

What is the algorithm to compute the optimal strategy for the prover?

- For a fixed verifier V and an input x, we represent all reactions of the verifier and all provers on a tree.
- The reaction of verifier depends on random bits
- A node at depth d is specified by the first d messages in the interaction (transcript of interaction at that point)
- The tree has polynomial depth, because verifier terminates in polynomial time.

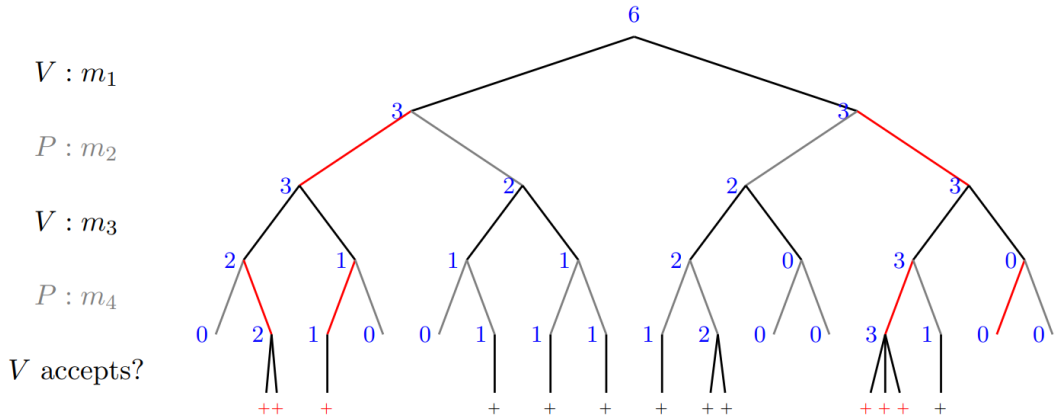

Figure 2: Tree of possible bitwise interaction for a fixed verifier and a fixed input x. Every prover defines a subtree in this tree. If 3 bits of randomness are used, then optimal strategy has a probability of acceptance of 6/8

It is now easy to construct an optimal prover by induction.

- We need to select the subtree which has maximum number of accept leaves in it.
- We do this from bottom to top.
- We assign a value 1 to each accept leaf.
- At each level in the tree that corresponds to a message from the verifier, the value of the nodes are equal to the sum of the value of the children.
- At each level in the tree that corresponds to a message from the prover, the values of the nodes is the maximum of the children's value.
- By induction, one can show that the root value is the maximal number a strategy for the prover can have. And hence this determines the maximal probability for which the verifier accepts x.

$P_{opt} \in PSPACE$

- The tree is exponentially large.
- we do not need the whole tree to compute number at root
- depth of tree is polynomial
- we traverse the tree in a depth-first way
- We only need to store the values in all nodes of the current branch, and each such value can be stored in poly-space.

P_{opt} algorithm

With the assumptions made in last slide, we express P_{opt} with following algorithm

Algorithm 2: The optimal prover P_{opt} for verifier V

function $P_{opt}(w, w^{(1)}, \ldots, w^{(m)})$
forall $i \in \{1, 2, \ldots, m-1$ **do**
  $v^{(i)} \leftarrow P_{opt}(w, w^{(1)}, \ldots, w^{(i)})$
$p_0 \leftarrow \mathbb{P}_{acc}(w, w^{(1)}, \ldots, w^{(m)}, v^{(1)}, \ldots, v^{(m-1)}, 0)$
if $p_0 \geq \frac{2}{3}$ **then**
  return 0;
else
  return 1;

#3SAT $\in$ **IP**: Introduction

Theorem (Lund, Fortnow, Karloff, Nisan 1989)

$$\#3SAT \in \mathrm{IP}$$

- #3SAT is the problem of counting the number of assignments that satisfy a given 3-CNF.
- The class of such problems, which ask for the number of solutions to problems in NP, is called #**P**.
- #3SAT is #**P**-complete.
- It is known that #**P** contains the whole polynomial hierarchy.

General Scenario

Let $f(x_1, \ldots, x_n)$ be a polynomial of degree $d \leq \text{poly}(n)$ over $\mathbb{Z}_p$, where p is a prime number $\geq 2^{2n}$ known to both Arthur and Merlin. Given any input, Arthur can evaluate f.

Merlin tries to prove claims of the following kind to Arthur:

$$\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n) = c \mod p$$

Where c is some constant in $\mathbb{Z}_p$.

There exists an interactive proof system for the above scenario. Such a system can be used for #3SAT too, because #3SAT can be reduced to the above scenario by arithmetizing a given 3-CNF into a polynomial of the kind above.

Arithmetizing a 3-CNF works as follows:

- Convert each clause:
  - Replace any $x \vee y \vee z$ by $1 - (1-x)(1-y)(1-z)$.
  - Replace any $\overline{x}$ by $1 - x$.
- Multiply the arithmetized versions of all the clauses together.

Example. Consider the 3-CNF $\phi(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$. Using the above rules, this gets converted to the polynomial:

$$f(x_1, x_2, x_3) = (1 - (1 - x_1)(1 - x_2))(1 - x_1(1 - x_2)(1 - x_3))$$

This replacement ensures that the polynomial obtained always evaluates to the same value as the 3-CNF for any given assignment.
If the number of literals is n, the degree of the polynomial obtained is $O(n^3)$. This satisfies the degree criterion for the general scenario.

Key Idea

If $f(x_1, \ldots, x_n)$ is the polynomial obtained by arithmetizing a given 3-CNF ϕ, then the number of assignments that satisfy ϕ is exactly equal to:

$$\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n)$$

Thus, the interactive protocol for #3SAT starts by Merlin claiming that

$$\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n) = c \mod p$$

Which is another way of saying that ϕ has c satisfying assignments.

$$\textbf{Claim: } \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n) = c \mod p$$

- Merlin sends the prime $p \geq 2^{2n}$ to Arthur. Since primality testing is in **P**, Arthur can verify that p is indeed a prime.
- Arthur implicitly asks Merlin to send him the univariate polynomial obtained by summing over all possible values of $x_2, x_3 \ldots x_n$. That is, Arthur asks for the coefficients of the following polynomial:

$$g(x_1) = \sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, x_2, \ldots, x_n)$$

The degree of $g(x_1)$ is $\leq d$, which means it has polynomially many coefficients.

Claim: $\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n) = c \mod p$

- Merlin sends over a polynomial $g'(x_1)$, which he claims is what Arthur demanded. From the definition of $g(x_1)$, this new claim implies that

$$g'(0) + g'(1) = c$$

  Arthur checks this. If this turns out not to be true, Arthur immediately rejects all claims, and the proof fails.
- Even if the check passes, there is a possibility that $g' \neq g$.
- **Note:** if Merlin's initial claim was false, then he wouldn't send g itself, as that will get him caught immediately. On the other hand, if Merlin's initial claim was true, he will simply send g itself to convince Arthur.

$$\textbf{Claim: } \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n) = c \mod p$$

- To check that $g' = g$, Arthur picks a random $r \in \mathbb{Z}_p$, calculates $g'(r) = c_1$ and implicitly asks Merlin to prove that $g(r) = c_1$.
- Note that, if g' and g are actually different, then the probability that $g'(r) = g(r)$ is very low.
- By the definition of $g(x_1)$, this is equivalent to asking Merlin to prove that

$$\sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots \sum_{x_n=0}^{1} f_1(x_2, x_3, \ldots, x_n) = c_1 \mod p$$

  Where $f_1(x_2, x_3, \ldots, x_n) = f(r, x_2, x_3, \ldots, x_n)$.
- f_1 is a polynomial with $n-1$ variables and degree $\leq d$. The protocol recurses to prove this claim.

Base Case. There will be n total rounds in the interaction. The final round will involve proving a claim of the form

$$\sum_{x_n=0}^{1} f_{n-1}(x_n) = c_{n-1}$$

But this can be easily checked in polynomial time by Arthur himself, as $f_{n-1}(x_n)$ is a univariate polynomial with degree $\leq d$. There is no need for Merlin to be involved. If this base case also passes Arthur's test, then Arthur finally accepts Merlin's initial claim.

If Merlin's initial claim is true, then Arthur will always accept. At each step of the protocol, Merlin can send the actual g when Arthur demands it, and that will always pass Arthur's tests.

If Merlin's initial claim is false, then one of the following could happen.

1. Merlin is unable to find a $g' \neq g$ such that $g'(0) + g'(1) = c$, and consequently fails Arthur's check.
2. Merlin is able to find a $g' \neq g$ such that $g'(0) + g'(1) = c$, but Arthur chooses an $r \in \mathbb{Z}_p$ such that $g'(r) \neq g(r)$. So, Merlin is forced to try to prove another claim that is false.
3. Merlin is able to find a $g' \neq g$ such that $g'(0) + g'(1) = c$, and Arthur happens to choose an r such that $g'(r) = g(r)$. Merlin is then asked to prove a claim that is true. In this case, Merlin will succeed in cheating Arthur. The probability of this happening is *very* low, as we'll show next.

#3SAT $\in$ **IP**: Analysis

- Given that g and g' are univariate polynomials with degree $\leq d$, and $g' \neq g$, the number of inputs $r \in \mathbb{Z}_p$ for which they are equal is the same as the number of roots of the non-zero polynomial $g' - g$.
- Since the number of roots of any polynomial with degree $\leq d$ is at most d, the probability that g' and g agree on a randomly chosen r is

$$\Pr[(g' - g)(r) = 0] = \frac{d}{|\mathbb{Z}_p|} = \frac{d}{2^{2n}}$$

- A lying Merlin will succeed in cheating Arthur only if, in any one of the n rounds, it so happens that the r that Arthur chose is such that $g'(r) = g(r)$. Using the union bound over all the n rounds, the probability of this happening is at most

$$n \times \frac{d}{2^{2n}} \leq \frac{n}{2^n} \qquad \text{(for large enough } n\text{)}$$

Theorem (Shamir 1990)

$$\mathrm{IP} = \mathrm{PSPACE}$$

- Shamir used arithmetization to prove this result, hardly two weeks after LFKW had proved theirs.
- The proof presented here is a simplified version due to Sheng (1992).

TQBF$\in$ **IP**: Introduction

We have already shown that **IP** $\subseteq$ **PSPACE**. In order to show that **PSPACE** $\subseteq$ **IP**, we will show that a **PSPACE**-complete problem, TQBF, admits an interactive proof scheme.
Recall, TQBF is the set of all Quantified Boolean Formulas that are true.
A quantified boolean formula is an expression of the form

$$\Psi = \forall x_1 \exists x_2 \ldots Q_n x_n \ \phi(x_1, x_2, \ldots, x_n)$$

Where $Q_n \in \{\forall, \exists\}$. Without loss of generality, ϕ is a 3-CNF.

TQBF$\in$ **IP**: Setup

To arithmetize the QBF, we start by arithmetizing ϕ.

$$\Psi = \forall x_1 \exists x_2 \ldots Q_n x_n \; f(x_1, x_2, \ldots, x_n)$$

Where $f(x_1, \ldots, x_n)$ is a polynomial with degree $d = O(n^3)$.

TQBF$\in$ **IP**: Setup

To arithmetize $\forall$ and $\exists$, we introduce the following operators:

$$\prod_{x_n=0}^{1} f(x_1, \ldots, x_n) = f(x_1, \ldots, x_{n-1}, 0) \times f(x_1, \ldots, x_{n-1}, 1)$$

For $\forall$, which computes $\phi(x_1, \ldots, x_{n-1}, 0) \wedge \phi(x_1, \ldots, x_{n-1}, 1)$.

$$\coprod_{x_n=0}^{1} f(x_1, \ldots, x_n) = \quad f(x_1, \ldots, x_{n-1}, 0) + f(x_1, \ldots, x_{n-1}, 1)$$
$$- f(x_1, \ldots, x_{n-1}, 0) \times f(x_1, \ldots, x_{n-1}, 1)$$

For $\exists$, which computes $\phi(x_1, \ldots, x_{n-1}, 0) \vee \phi(x_1, \ldots, x_{n-1}, 1)$.

TQBF$\in$ **IP**: Setup

Ψ can now be rewritten as

$$\Psi = \forall x_1 \exists x_2 \dots Q_n x_n \; f(x_1, x_2, \dots, x_n)$$

$$\downarrow$$

$$\Psi = \prod_{x_1=0}^{1} \coprod_{x_2=0}^{1} \dots \bigsqcup_{x_n=0}^{1} f(x_1, x_2, \dots, x_n)$$

Where $\bigsqcup \in \{\prod, \coprod\}$.

- We could now try to apply the same protocol we used in the case of #3SAT.
- But there is a big issue: Each time we apply $\prod$ or $\coprod$, we end up doubling the degree of the polynomial!
- As a result, the univariate polynomial that Arthur requests may have exponentially many coeffecients, and so Merlin will not be able to send it to him.

TQBF$\in$ **IP**: Setup

Idea. If we consider only $0/1$ inputs, a polynomial evaluates to the same value if we replace all x^k by just x. For example,

$$3x_1^5x_2^2 + x_1^2 + x_2 \text{ can be replaced with } 3x_1x_2 + x_1 + x_2$$

We define the following linearization operation.

$$L_n f(x_1, \ldots, x_n) = x_n \times f(x_1, \ldots, x_{n-1}, 1) + (1 - x_n) \times f(x_1, \ldots, x_{n-1}, 0)$$

Now, we can sprinkle these linearization operators in the arithmetization of Ψ to keep the degree in check. We rewrite Ψ as

$$\Psi = \prod_{x_1=0}^{1} L_1 \coprod_{x_2=0}^{1} L_1L_2 \prod_{x_3=0}^{1} \ldots \bigsqcup_{x_n=0}^{1} L_1L_2 \ldots L_n \; f(x_1, \ldots, x_n)$$

TQBF$\in$ **IP**: Setup

In order to prove that $\Psi \in$ TQBF, Merlin tries to prove the following claim to Arthur:

$$\prod_{x_1=0}^{1} L_1 \coprod_{x_2=0}^{1} L_1 L_2 \prod_{x_3=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 L_2 \dots L_n \, f(x_1, \dots, x_n) = 1$$

There are a total of $n(n+3)/2$ operators on the left side. Each round, Merlin helps Arthur "strip" one operator.
The interaction is very similar to that for #3SAT $\in$ **IP**. The only difference is that now, instead of just $\sum$, we are dealing with 3 operators, which give rise to three cases.

Claim: $\prod_{x_i=0}^{1} L_1 \dots L_i \coprod_{x_{i+1}=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_{i-1}, x_i, \dots, x_n) = c \mod p$

- Arthur implicitly demands the following univariate polynomial:

$$g(x_i) = L_1 \dots L_i \coprod_{x_{i+1}=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_{i-1}, x_i, \dots, x_n)$$

- In response, Merlin sends the polynomial $g'(x_i)$. Arthur performs the check:

$$\prod_{x_i=0}^{1} g'(x_i) = c \mod p$$

  If the check fails, Arthur rejects.
- If the check passes, Arthur picks a random $r_i \in \mathbb{Z}_p$, calculates $c' = g'(r_i)$ and asks Merlin to prove that $g(r_i) = c'$.

Claim: $\prod_{x_i=0}^{1} L_1 \dots L_i \coprod_{x_{i+1}=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_{i-1}, x_i, \dots, x_n) = c \mod p$

- The claim $g(r_i) = c'$ translates to

$$L_1 \dots L_i \coprod_{x_{i+1}=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_{i-1}, r_i, \dots, x_n) = c' \mod p$$

  The protocol recurses to prove this claim.
- The second case is when Merlin tries to prove that:

$$\coprod_{x_i=0}^{1} L_1 \dots L_i \prod_{x_{i+1}=0}^{1} \dots \bigsqcup_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_{i-1}, x_i, \dots, x_n) = c \mod p$$

  This case is handled in a very similar fashion.

TQBF$\in$ **IP**: Case 3

Claim: $L_i L_{i+1} \dots L_j \prod_{x_{j+1}=0}^{1} \dots \prod_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_i, \dots, r_j, x_{j+1}, \dots x_n) = c \mod p$

- This time, Arthur demands a different kind of univariate polynomial:

$$g(x_i) = L_{i+1} \dots L_j \prod_{x_{j+1}=0}^{1} \dots \prod_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, x_i, \dots, r_j, x_{j+1}, \dots x_n)$$

- Merlin sends over $g'(x_i)$. Arthur checks that $(L_i g'(x_i))(r_i) = c$. If the check fails, Arthur rejects.
- If the check passes, Arthur selects a new $r_i' \in \mathbb{Z}_p$, calculates $c' = g'(r_i')$ and asks Merlin to prove that $g(r_i') = c$. The new claim is

$$L_{i+1} \dots L_j \prod_{x_{j+1}=0}^{1} \dots \prod_{x_n=0}^{1} L_1 \dots L_n f(r_1, \dots, r_i', \dots, r_j, x_{j+1}, \dots x_n) = c' \mod p$$

Observation. The degree of the polynomials that Merlin sends during the interaction is at most 2, except for the final set of linearization operators.

- For case 1 and 2, the degree is exactly 1, because of the linearization operators that follow.
- For case 3, the degree may be 2, because of the lack of linearization for some variables.

If Merlin's initial claim was false, then for case 1 and 2, the probability that Merlin gets lucky and Arthur chooses an r_i such that $g'(r_i) = g(r_i)$ even though $g'(x_i) \neq g(x_i)$ is $\frac{1}{|\mathbb{Z}_p|} = \frac{1}{p}$.
For the inner linearization operators, this probability is at most $\frac{2}{p}$, while for the final set of linearization operators, it is at most $\frac{3m}{p}$, where m is the number of clauses in ϕ.

TQBF$\in$ **IP**: Analysis

Applying union bound, the probabilty that Merlin successfully cheats Arthur is

$$n \times \frac{1}{p} + n \times \frac{3m}{p} + \frac{2}{p} \times \sum_{i=1}^{n-1} i = \frac{3mn + n^2}{p}$$

A polynomially large p will suffice to keep this probability low.

References

1. http://www.villesalo.com/article/IeP.pdf
2. https://theory.cs.princeton.edu/complexity/book.pdf
3. https://www.hse.ru/mirror/pubs/share/206277369
4. https://www.cs.cmu.edu/∼odonnell/complexity17/odonnell-graduate-complexity-notes.pdf
5. http://www.cs.umd.edu/∼jkatz/complexity/f11/lecture19.pdf
6. Babai, L., E-Mail And The Unexpected Power Of Interaction. Available at: https://www.cs.princeton.edu/courses/archive/spring09/cos522/BabaiEmail.pdf.