

# The Proof of $IP = PSPACE$

Larisse D. Voufo

March 29th, 2007

For a long time, the question of how a verifier can be convinced with high probability that a given theorem is provable without showing the whole proof, and of how rapidly this can be done, remained an open problem. This led interested parties to formulate and extensively study it in terms of “*interactive protocols*”.

In 1992, Adi Shamir surprisingly completed the proof of  $IP = PSPACE$ , allowing  $IP$  to be placed in the standard classification of feasible computations[2]. This proof amazingly 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”. [1]

This paper focusses on laying out that proof, but first introduces some preliminary knowledge on quantified boolean formulae.

## Contents

<b>1 Overview: Interactive Proof Systems and <math>IP = PSPACE</math></b>	<b>1</b>
<b>2 Preliminary Notes on Quantified Boolean Formulae</b>	<b>3</b>
<b>3 Simulating an Interactive Proof System by a PSPACE machine</b>	<b>5</b>
<b>4 An Interactive Proof System for Deciding the Truth of QBFs</b>	<b>8</b>
<b>5 Conclusion</b>	<b>10</b>

## 1 Overview: Interactive Proof Systems and $IP = PSPACE$

Interactive Proof Systems (IPS) were first introduced in 1985 by Goldwasser et al.[3] They consist of an all-powerful (unbounded computational resources) Turing Machine (TM) called a prover ( $P$ ), and of a probabilistic polynomial-time machine called a Verifier ( $V$ ).  $V$  has access to a random bit string not visible to  $P$ , and whose length is polynomial on the input size. In addition, both  $P$  and  $V$  originally receive the same input string. Then they engage into a series of question-answers where  $P$  continuously presents a proof that an input  $w$  is in some language  $L$ , and  $V$  checks that the presented proof is in fact correct. The interaction completes after a polynomial number of messages (relative to the input size), and  $V$  must

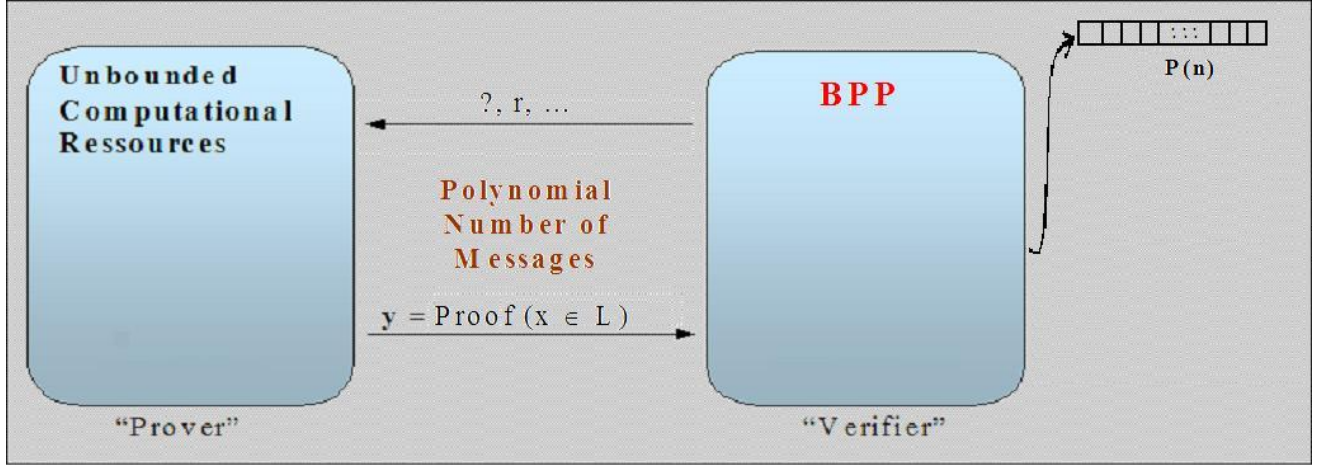


Figure 1: A graphical description of an Interactive Proof System

decide whether or not  $w$  is in the language with only a  $1/3$  chance of error. Of course,  $V$  must also ensure that it is not fooled by a dishonest (lying) Prover( $Q$ ) in the process.

The above introduces the complexity class called  $IP$ . More formally, a problem is in the class  $IP$  if it is solvable by an Interactive Proof System. By definition:

For any language  $L$ ,  $L \in IP \Leftrightarrow \exists V, P | \forall Q, w$ :

- ▷  $w \in L \Rightarrow Pr[V \leftrightarrow P \text{ accepts } w] \geq 2/3^1$
- ▷  $w \notin L \Rightarrow Pr[V \leftrightarrow Q \text{ accepts } w] \leq 1/3$

The complexity class  $PSPACE$  defines all problems that can be solved in polynomial space but may require exponential time.

**The proof in question:** The proof for  $IP = PSPACE$  consists of two main subproofs:  $IP \subseteq PSPACE$  and  $PSPACE \subseteq IP$ .

Proving that  $IP \subseteq PSPACE$  can be trivially done by presenting a simulation of an IPS by a polynomial space machine.

The proof for  $PSPACE \subseteq IP$  is credited to Adi Shamir[1]<sup>2</sup> who presented a reduction of the problem into: finding a  $PSPACE$ -complete problem, and then proving that that problem is in  $IP$ . This said, we realize that the problem of deciding the truth of Quantified Boolean Formulae (QBF) has been known to be  $PSPACE$ -Complete. Hence, showing that it is in  $IP$  will constitute our proof. In other words, let's represent the problem of deciding the truth of QBF with  $TQBF$ . Then,  $(TQBF \in PSPACE - Complete)$  and  $(TQBF \in IP) \Rightarrow PSPACE \subseteq IP$ .

We can show that  $TQBF \in IP$  simply by illustrating the existence of an interactive proof system for solving  $TQBF$ . However, a major part of that proof relies on understanding

<sup>1</sup>It is important to notice that, more formally, the delimiting constant (currently " $2/3$ ") should be " $1/2+\delta$ ", where  $\delta > 0$ . It just happens that " $2/3$ " is the most commonly used one.

<sup>2</sup>Notes on QBF and  $PSPACE \subseteq IP$  reuse a fair amount of phrases from his paper, cited in this paper, for lack of a better way to put them.

some operations on QBF as defined in Adi Shamir's "IP = PSPACE" paper[1]. Therefore, the rest of this paper consist of three parts. First, we will present an overview on QBF. Then, we will simulate an IPS using a PSPACE machine, and last but not the least, we will present an IPS for TQBF.

## 2 Preliminary Notes on Quantified Boolean Formulae

- ▷ A **Quantified Boolean Formula** is the closure of the set of Boolean variables  $x$ , and their negations  $\bar{x}$ , under the operations  $\wedge$ (and),  $\vee$ (or),  $\forall$ (universal quantification), and  $\exists$ (existential quantification).
- ▷ A **closed QBF** is a QBF in which all the variables are quantified. It can be evaluated to either  $T$ (true) or  $F$ (false).
- ▷ An **Open QBF with  $k > 0$  free variables** can be interpreted as a boolean function from  $\{T, F\}^k$  to  $\{T, F\}$ .
- ▷ A **closed QBF** is called **simple** if in the given syntactic representation, every occurrence of each variable is separated from its point of quantification by at most one universal quantifier (and arbitrarily many other symbols).

*Example:*

$\forall x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge \forall x_4 (x_2 \wedge x_3 \wedge x_4)]$  is simple.

However,  $\forall x_1 \forall x_2 [(x_1 \wedge x_2) \wedge \forall x_3 (\bar{x}_1 \wedge x_3)]$  is not simple, due to an extra quantifier between  $\forall x_1$  and the usage of  $x_1$  in  $(\bar{x}_1 \wedge x_3)$ .

- ▷ Notice that every QBF of size  $n$  can be transformed into an equivalent simple QBF whose size is polynomial in  $n$ .

*Example:*

$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \dots Q(x_1, x_2, x_3, \dots)$ , where  $Q$  is quantifier free, can be transformed into:

$\exists x_1^0 \forall x_2^0 \exists x_1^1 (x_1^1 = x_1^0) \wedge \exists x_3^0 \forall x_4^0 \exists x_1^2 \exists x_2^1 \exists x_3^1 (x_1^2 = x_1^1) \wedge (x_2^1 = x_2^0) \wedge (x_3^1 = x_3^0) \wedge \exists x_5^0 \dots Q(x_1^{j_1}, x_2^{j_2}, x_3^{j_3}, \dots)$ ,

where  $x_i^{j_i}$  is the last name given to the initial Boolean variable  $x_i^0$ .

This QBF is simple by definition, and contains a quadratic number of variables (compared to the original QBF), since each one of the original  $n$  variables is being renamed under a linear bound on  $n$ .

### Arithmetizing QBF

- ▷ The **arithmetization of a given QBF** is its transformation into an arithmetic form, under the following scheme:

- ▶ Boolean variable  $x_i \longrightarrow$  new variable  $z_i$ ,

- ▶  $\bar{x}_i \longrightarrow (1 - z_i)$
- ▶  $\wedge, \vee, \forall x, \exists x \longrightarrow *, +, \prod_{z_i \in \{0,1\}}, \sum_{z_i \in \{0,1\}}$ ; respectively.

*Running example:*

Let  $B = \forall x_1 [\bar{x}_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2) \vee x_3]$ ,

Its arithmetic form is:  $A = \prod_{z_1 \in \{0,1\}} [(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2) + z_3]$ , with value 2.

- ▷ Notice that a closed QBF is henceforth true iff the value of its arithmetic form is nonzero.
- ▷ However, this value can get quite large and has an upper bound of  $O(2^{2^n})$ .

As an illustration, notice that for any (potentially open) subexpression  $B$  of a given QBF, its maximal value  $v$ , under all possible 0/1 substitutions to the free variables, satisfies the following (where  $v'$  and  $v''$  are the maximal values of  $B'$  and  $B''$  respectively):

- ▶ If  $B$  is  $x_i$  or  $\bar{x}_i$ , then  $v = 1$ .
- ▶ If  $B$  is  $B' \vee B''$ , then  $v \leq v' + v''$ .
- ▶ If  $B$  is  $B' \wedge B''$ , then  $v \leq v' \cdot v''$ .
- ▶ If  $B$  is  $\exists x_i B'$ , then  $v \leq 2v'$ .
- ▶ If  $B$  is  $\forall x_i B'$ , then  $v \leq v'^2$ .
- ▶ If  $B$  is closed, then  $v$  coincides with the value of its arithmetic form.

Thus, the worst case scenario happens when the given QBF is constituted of  $B$ , with the simplest operation between variables, nested within a succession of  $\forall$  clauses as in the formula:  $\forall x_1 \forall x_2 \forall x_3 \dots \forall x_n B$ .

Given that each universal quantifier squares the previous value, the maximal value of this formula is:  $((((v^2)^2)^2) \dots)^2 = v^{2 \cdot 2 \cdot 2 \dots 2} = v^{2^n}$ .

Now, out of the remaining operations (excluding  $\forall$ ), substituting  $v'$  and  $v''$  by their maximal variable value of 1 results in  $v = 2$ ; hence producing an upper bound of  $O(2^{2^n})$

- ▷ Since  $V$  cannot handle such large numbers, Modulo Arithmetic can be used to reduce them modulo some smaller prime  $p$ . The idea is that given the arithmetic value  $A$  of a closed QBF  $B$ , there exists a prime  $p$  of length polynomial in the size of  $B$ , such that  $A \neq 0 \pmod{p}$  iff  $B$  is true.

**Functionalizing QBF** Given a closed arithmetic expression  $A$ ,

- ▷ The **functional form**  $A'$ , of  $A$ , is computed by eliminating the leftmost  $\prod_{z_i \in \{0,1\}}$ , or  $\sum_{z_i \in \{0,1\}}$ ; and considering  $A'$  as a polynomial function  $q(z_i)$  of one free variable  $z_i$ .

*Running example:*

The functional form of  $A$  is:  $A' = [(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2) + z_3]$ .

By evaluating all the summations and products,  $P$  can express it as the polynomial  $q(z_1) = z_1^2 + 1$ .

- ▷ The **randomized form** of  $A$  is  $A'(z_i = r)$  in which  $z_i$  is set to a random number  $r$  modulo  $p$  supplied by the verifier.

This randomized form can again be evaluated to a constant, but the number of  $\prod$  and  $\sum$  symbols is reduced by 1. Note that the simplicity of  $A$  is preserved by these transformations.

*Running example:*

The randomized form of  $q(z_1)$  with  $z_1 = 3$  is:  $A'(z_1 = 3) = [(1-3) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (3 \cdot z_2) + z_3]$ .

Its value 10 can be deduced from applying 3 to the function  $q(z_1)$ .

- ▷ Notice that the degree of  $q(z_1)$  can be exponentially high, due to products. However, we can reduce it by simplifying the original expressions.

The idea is that if a QBF is simple, then the degree of the polynomial  $q(z_1)$  that describes the functional form of its arithmetic form grows at most linearly with the size of the QBF. Also, by adding sufficiently many dummy variables to the quantifier-free sub-expressions of the QBF, this degree can be reduced to 3, and thus allowing  $q(z_1)$  to be represented by just four prime numbers.

### 3 Simulating an Interactive Proof System by a PSPACE machine

- ▷ Consider  $A$ , a language in  $IP$ .

- ▷ By the definition of an  $IP$  system, we know that:

On input  $w$  with length  $n$ ,  $A$ 's verifier  $V$  exchanges exactly  $p = f(n)$  messages, based on a random number  $r$  gotten from its random string, and given a function  $f$  that is polynomial in  $n$ .

- ▷ This said, let's **construct a PSPACE machine  $M$  that simulates  $V$** .

To do this, we define  $M$  as follows:

$$\Pr[M \text{ accepts } w] = \max_p \Pr[V \leftrightarrow P \text{ accepts } w]$$

That is, the probability that  $M$  accepts an input  $w$  corresponds to the maximum of all the probabilities that the interaction of  $V$  and  $P$  would accept  $w$  given a possible value of  $r$ . Remember that there are  $p$  possible values for  $r$  ( $p = \text{size of } r$ ).

- ▷ By the definition of  $IP$ ,
  - $\Pr [M \text{ accepts } w] \geq 2/3$  if  $w \in L$ .
  - $\Pr [M \text{ accepts } w] \leq 1/3$  if  $w \notin L$ .
- ▷ Now, let's **verify that the value of  $\Pr [M \text{ accepts } w]$  can in fact be calculated in PSPACE.**

- ▶ Let  $M_j$  be the sequence of messages,  $m_1\#m_2\#\dots\#m_j$ , exchanged by  $P$  and  $V$ .
- ▶ And let's **generalize the interaction of  $V$  and  $P$  to start with an arbitrary message stream  $M_j$**  as follows:
  - $(V \leftrightarrow P)(w, r, M_j) = \text{accept}$ , if  $M_j$  can be extended with the messages  $m_{j+1}$  through  $m_p$  such that:
    1. For  $j \leq i < p$ , where  $i$  is even,  $V(w, r, M_j) = m_{i+1}$  ( $\Rightarrow V$  sends message)
    2. For  $j \leq i < p$ , where  $i$  is odd,  $P(w, r, M_j) = m_{i+1}$  ( $\Rightarrow P$  sends message)
    3. The final message  $m_p$  in the message history is *accept* (final message is to *accept*)

Notice that items (1) and (2) ensure the validity of  $M_j$  and item (3) ensures that  $M_j$  leads to an accepting state)

- ▶ From this, we gather the following generalization:
  - $\Pr [V \leftrightarrow P \text{ accepts } w \text{ starting at } M_j] = \Pr [(V \leftrightarrow P)(w, r, M_j) = \text{accept}]$
- ▶ **Generalizing the earlier definition of  $M$  further**, we get:
  - $\Pr [M \text{ accepts } w \text{ starting at } M_j] = \max_p \Pr [V \leftrightarrow P \text{ accepts } w \text{ starting at } M_j]$

Therefore, our verification step now consists of proving that the value of

$\Pr [M \text{ accepts } w \text{ starting at } M_j]$  can be computed in *PSPACE*.

- ▷ To do that, let's **define the function  $N_{M_j}$ , for every  $0 \leq j \leq p$  and every message history  $M_j$ <sup>3</sup>**, such that:

$$N_{M_j} = \begin{cases} 0 & \text{if } j = p \text{ and } m_p = \text{reject}. \\ 1 & \text{if } j = p \text{ and } m_p = \text{accept}. \\ \max_{m_{j+1}} N_{M_{j+1}} & \text{if } j < p \text{ and } j \text{ is odd.} \\ wt - \text{avg}_{m_{j+1}} N_{M_{j+1}} & \text{if } j < p \text{ and } j \text{ is even.} \end{cases},$$

where:

- ▶ Let  $\Pr_r =$  probability taken over a particular random value  $r$ .

---

<sup>3</sup>Notice that a probability at  $M_0$  means a probability before starting to "read" the  $p$  messages.

- ▶ Then,  $wt-avg_{m_{j+1}} N_{M_{j+1}} = \sum_{m_{j+1}} (\Pr_r [V(x, r, M_j) = m_{j+1}] N_{M_{j+1}})$  = the average of  $N_{M_{j+1}}$ , weighted by the probability that  $V$  sends message  $m_{j+1}$ .

To better understand this, remember that in an  $IP$  system,

- ▶ On a given input,  $V$  sends and receives a polynomial number of messages to and from  $P$ , each time with a probability that depends on the randomly chosen  $r$ .
  - ▶  $V$  makes a decision based on the maximum of all the probabilities of accepting messages. Therefore, the only way for  $P$  to ensure that its message is accepted is by making its probability equal that maximum probability.
  - ▶ Alternatively,  $P$  makes a decision based on averaging all the probabilities of receiving messages<sup>4</sup>. Hence, in order for  $V$  to ensure that its message is accepted, it has to ensure that its probability is equal to that average.
- ▷ We notice that any attempt to layout the “values” of  $N_{M_j}$  suggests that it describes a  $PSPACE$  machine. However, to be certain of that, let’s consider  $M_0$ , and **show that  $N_{M_0}$  can be computed in polynomial space.**
- This entails simply noticing that, to compute  $N_{M_0}$ , an algorithm can recursively calculate the values  $N_{M_j}$  for every  $j$  in  $M_j$ . Moreover, since the depth of the recursion is  $p$ , only polynomial space is necessary.
- ▷ Now that we know that  $N_{M_j}$  does in fact describe a  $PSPACE$  machine, let’s **verify that the value of  $\Pr [M \text{ accepts } w \text{ starting at } M_j]$ , and thus that of  $\Pr [M \text{ accepts } w]$ , can be computed in  $PSPACE$ , through  $N_{M_j}$ .**

▷ This entails showing that:  $N_{M_0} = \Pr [M \text{ accepts } w]$ .

▷ For that, we must show the following:

For every  $0 \leq j \leq p$ , and every  $M_j$ ,  $N_{M_j} = \Pr [M \text{ accepts } w \text{ starting at } M_j]$

▷ **Proof by induction on  $j$ :**

▶ **Base case:**  $j = p$

\*  $m_p$  is either *accept* or *reject*.

\* If  $m_p$  is *accept*,

then  $N_{M_j} = 1$  by definition of  $N_{M_j}$ ,

and  $\Pr [M \text{ accepts } w \text{ starting at } M_j] = 1$  since the message stream indicates acceptance.

\* A similar argument holds for the case where  $m_p$  is *reject*.

▶ **Induction Hypothesis (on  $k \geq 0$ ):**

$k = j + 1 \leq p$ ,

and  $N_{M_{j+1}} = \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}]$ .

---

<sup>4</sup>Notice that this operation (average) could be anything since the  $P$  is set up to accept any message from  $V$ . “Averaging” simply works as means of “dont-care” on inputs from  $V$ .

► **Prove true at  $k - 1$ :**

$$N_{M_j} = \Pr [M \text{ accepts } w \text{ starting at } M_j]$$

\* If  $j$  is even, then:

- $m_{j+1}$  is a message from  $P$  to  $V$ .
- By def. of  $N_{M_j}$ ,  $N_{M_j} = \sum_{m_{j+1}} (\Pr_r [V(x, r, M_j) = m_{j+1}] N_{M_{j+1}})$ .
- By IH,  $N_{M_j} = \sum_{m_{j+1}} (\Pr_r [V(x, r, M_j) = m_{j+1}] \cdot \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}])$ .
- By def.,  $N_{M_j} = \Pr [M \text{ accepts } w \text{ starting at } M_j]$ .

\* If  $j$  is odd, then:

- $m_{j+1}$  is a message from  $V$  to  $P$ .
- By def. of  $N_{M_j}$ ,  $N_{M_j} = \max_{m_{j+1}} N_{M_{j+1}}$ .
- By IH,  $N_{M_j} = \max_{m_{j+1}} \cdot \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}]$ .
- $N_{M_j} = \Pr [M \text{ accepts } w \text{ starting at } M_j]$ , since:
  1.  $\max_{m_{j+1}} \cdot \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}] \leq \Pr [M \text{ accepts } w \text{ starting at } M_j]$ , since the prover on the right-hand side could send the message  $m_{j+1}$  to maximize the expression on the left-hand side.
  2.  $\max_{m_{j+1}} \cdot \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}] \geq \Pr [M \text{ accepts } w \text{ starting at } M_j]$ , since the same prover cannot do any better than send that same message.

From the above, we get:

$$\max_{m_{j+1}} \cdot \Pr [M \text{ accepts } w \text{ starting at } M_{j+1}] = \Pr [M \text{ accepts } w \text{ starting at } M_j]$$

So,  $N_{M_j} = \Pr [M \text{ accepts } w \text{ starting at } M_j]$  and  $N_{M_j}$  is a *PSPACE* machine.

▷ Therefore,  $\Pr [M \text{ accepts } w \text{ starting at } M_j]$  is a *PSPACE* machine.

▷ Thus,  $\Pr [M \text{ accepts } w]$  is a *PSPACE* machine.

▷ So, the machine  $M$  does simulate an  $V$  in an *IPS*.

Hence,  $IP \subseteq PSPACE$ .

## 4 An Interactive Proof System for Deciding the Truth of QBFS

Given the arithmetic form  $A$  of a simple QBF  $B$ , We want to prove that  $A \neq 0 \pmod{p}$  for some polynomially long prime  $p$ . For this purpose, there are three important steps:

1. **First step of the interactive proof:**

$P$  chooses  $p$  and sends it to  $V$ , along with a written proof of primality, since even an infinitely powerful cheating prover cannot find such a prime if  $A = 0$ .

**2. During the next steps of the interaction protocol:**

$V$  randomly chooses  $p$ , using the fact that for most  $p$ ,  $A \neq 0 \pmod{p}$  iff  $B$  is true.

- ▷ Notice that one can prove both membership and non-membership by the same protocol.
- ▷ However, if the chosen  $p$  happens to be a divisor of  $A$ , even an honest prover may be unable to prove a correct statement. Therefore, this protocol has imperfect completeness.

Recall what we saw in part 2 about the exponentially high value of the degree of the polynomial  $q(z_i)$  and how we can improve it by simplifying the original QBF.

**3. The very simple interactive protocol for proving that  $A \neq 0 \pmod{p}$ .**

- ▷  $P$  sends the claimed value  $a$  of  $A \pmod{p}$  to  $V$ , and justifies this claim by considering successively smaller subexpressions of  $A$ .
- ▷ At any intermediate stage of the protocol, the current expression  $A$  is split into  $A_1 + A_2$  or  $A_1 \cdot A_2$ , where  $A_1$  is a polynomial with fully instantiated variables (whose value  $a_1$  can be computed by  $V$  himself), and  $A_2$  starts with the leftmost  $\Pi$  or  $\Sigma$  symbol of  $A$ .
- ▷  $P$  and  $V$  then repeatedly execute the following simplification steps:
  - (a) If  $A_2$  is *empty*,  $V$  stops and accepts the claim iff  $a = a_1$ .
  - (b) If  $A_1$  is *nonempty*,  $V$  replaces  $A$  by  $A_2$ , and replaces  $a$  by  $(a - a_1) \pmod{p}$  or  $(a/a_1) \pmod{p}$  (depending on the operator that connects  $A_1$  and  $A_2$ ).  
If  $V$  tries to divide  $a$  by  $a_1 = 0 \pmod{p}$ , he stops and accepts the claim iff  $a = 0 \pmod{p}$ .
- ▷ Otherwise,
  - ▶  $P$  sends the polynomial descriptions  $q(z_i)$  of  $A'$  to  $V$ .
  - ▶  $V$  checks that  $a = (q(0) + q(l)) \pmod{p}$  or  $a = (q(0) \cdot q(l)) \pmod{p}$  (depending on the first symbol of  $A_2$ ), sends a random  $r \in Z_P$  to  $P$ , replaces  $A$  by  $(A'(z_i = r)) \pmod{p}$ , and replaces  $a$  by  $q(r) \pmod{p}$ .

With this protocol, we notice two main things:

1. When  $B$  is true and  $P$  is honest,  $V$  always accepts the proof.
2. When  $B$  is false,  $V$  accepts the proof with negligible probability.

PROOF :

1. An honest  $P$  can always justify his claimed values and polynomials.

2. A cheating prover who supplied an incorrect value of  $a$  must provide an incorrect polynomial  $q(zi)$  to support his claim, since  $a$  is checked against  $q(0) + q(l)(\text{mod } p)$  or  $q(0) \cdot q(l)(\text{mod } p)$ . By the interpolation theorem, such an incorrect polynomial of degree  $t$  can agree with the correct polynomial on at most  $t$  of the  $p$  points in  $Z_P$ . When the value of  $t$  is a polynomial and the value of  $p$  is exponential in the size of  $B$ , there is only a negligible probability that the incorrect  $q$  yields a correct value when evaluated at a random point  $r$  chosen by  $V$ . As a result, a cheating  $P$  is forced to provide incorrect values for successively smaller sub-expressions, until he is exposed with overwhelming probability when  $V$  evaluates the final sub-expression by himself.

**Additional remarks:** In his paper, Adi Shamir states that “a single application of this protocol suffices to make the probability of cheating exponentially small, and there is no need to iterate it as in other interactive proofs. However, the protocol seems to be inherently sequential, and it is a major open problem whether it can be executed with a small (e.g., logarithmic) number of rounds. Note that the existence of a constant round protocol for  $IP$  would collapse the polynomial hierarchy to its second level...”[1].

**Weak Verifiers can improve the space-bound required by the protocol** He also points out the fact that this interactive protocol requires polynomial time and polynomial space verifiers, and demonstrates a way to greatly improve the space bound using a **weak verifier**, which is characterized by the following:

- ▷ Its running-time is polynomial,
- ▷ Its workspace is logarithmic,
- ▷ It has a two-way read-only access to a random tape, and
- ▷ Its messages consist solely of the random bits it reads.

## 5 Conclusion

We have presented the complete proof of  $IP = PSPACE$  in different terms than most current literatures. This will hopefully improve one’s understanding of this proof.

It is important to notice, though, that this proof not only placed the  $IP$  class in the standard classification of feasible computations, but also provided alternative definitions of classical (and quantum) complexity classes<sup>5</sup>.

---

<sup>5</sup>See the following for a brief overview:

- ▷ Voufo. 2006. "Quantum Complexities and Interactive Proof Systems". [<http://www.cs.indiana.edu/~lvoufo/project.pdf>]
- ▷ Voufo. 2006. "Quantum Complexity Classes". [<http://www.cs.indiana.edu/~lvoufo/qcc.pdf>]

## References

- [1] Shamir, A. 1992. *IP = PSPACE*. J. ACM 39, 4 (Oct. 1992), 869-877. DOI=<http://doi.acm.org/10.1145/146585.146609>
- [2] Hartmanis, J., R. Chang, D. Ranjan, P. Rohatgi. 1990. *On IP = PSPACE and Theorems with Narrow Proofs*.
- [3] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. *The Knowledge complexity of interactive proof-systems*. Proceedings of 17th ACM Symposium on the Theory of Computation, Providence, Rhode Island. 1985, pp. 291-304.