IP = PSPACE

Johannes Mittmann

Technische Universität München

May 21, 2006

Abstract

In [Sh92], Adi Shamir proved a complete characterization of the complexity class **IP**. He 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 within polynomial space.

This paper gives a detailed description of the proof. Besides the original paper, it is based on [Pa94] and [SchPr98].

Contents

1 Introduction

Interactive proof systems were introduced by Goldwasser *et al.* in [GMR85]. An earlier result by Papadimitriou [Pa83] implied that $\mathbf{IP} \subseteq \mathbf{PSPACE}$. But **IP** was considered to be only a slight generalization of **NP** and it was not even expected to contain **coNP**. Oracle results actually suggested the converse, what shows that the proof presented in this paper does not "relativize".

However, the situation changed on November 27, 1989, when Nisan surprised a few insiders by e-mail announcement (see [Ba90]) that he had found a multi-prover interactive protocol for the permanent. This result was improved soon by Lund, Fortnow and Karloff to the (single-prover) LFKN-protocol for the permanent [LFKN92], and announced on the mailing list on December 13, 1989. Because of results of Valiant [Va79] (the permanent is #**P**-complete) and Toda [To89] ($\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$), this implied that **IP** contains the whole polynomial hierarchy $\mathbf{PH} \subseteq \mathbf{IP}$. Now, **IP** was much more powerful than previously suspected, so why could not **IP** actually contain **PSPACE**?

This question was finally answered on December 26, 1989. The proof was accomplished by extending the techniques of Lund *et al.* to quantified Boolean formulas. A difficulty that arised was the exponential growth of the degree of polynomials involved in arithmetizations of those formulas. The first one who could overcome this problem was the Indian Adi Shamir who worked over the Christmas holidays.

2 Polynomial Space

The complexity class

$$\mathbf{PSPACE} = \bigcup_{k>0} \mathbf{SPACE}(n^k)$$

is the class of problems that are decided by a deterministic Turing machine using a polynomial amount of space. We review some of the facts about **PSPACE**.

There are two results about space whose analogous statements about time are widely believed not to be true. The first one is Savitch's Theorem which states that $\mathbf{PSPACE} = \mathbf{NPSPACE}$. The second is the Immerman-Szelepscényi Theorem which says that nondeterministic space classes are closed under complement. The relationship to other complexity classes is given by the following tower of inclusions:

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}.$$

From the Space Hierarchy Theorem we obtain that the inclusion $\mathbf{L} \subsetneq \mathbf{PSPACE}$ is proper, but there is nothing known about the four inclusions in between.

In this paper we want to show the characterization **IP** = **PSPACE**. In order to proof the identity of complexity classes (that are closed under reductions), it suffices to show that they share the same complete problems. Therefore we introduce a decision problem that is complete for **PSPACE**.

2.1 Quantified Satisfiability

Satisfiability for Boolean formulas is **NP**-complete. It turns out that, if we add quantifiers to our formulas, satisfiability becomes **PSPACE**-complete.

Definition 1. Let $X = \{x_1, x_2, \dots\}$ be an alphabet of *Boolean variables*. They can take the two *truth values* **true** and **false**.

A *quantified Boolean expression (QBF)* ϕ is defined inductively by

1. a Boolean variable x_i,

or an expression of the form

2. $\neg\phi_1$ (*negation*),
3. $\phi_1 \vee \phi_2$ (*disjunction*),
4. $\phi_1 \wedge \phi_2$ (*conjunction*),
5. $\exists x_i\, \phi_1$ (*existential quantification*),
6. $\forall x_i\, \phi_1$ (*universal quantification*),

where ϕ_1 and ϕ_2 are quantified Boolean expressions.

As the definition is inductive, many proofs about QBFs will work by induction on the structure of the formula.

Let ϕ be a QBF. A variable x_i in ϕ that is not quantified is called *free*. If ϕ does not have any free variables, it is called *closed*. A closed QBF evaluates to either **true** or **false**.

Two QBFs ϕ, ψ are *equivalent*, written $\phi \equiv \psi$, if for any truth assignment, ϕ is satisfied if and only if ψ is satisfied. Besides the obvious commutativity and associativity laws for $\vee$ and $\wedge$, the following properties hold.

Proposition 2. *Let ϕ and ψ be QBFs. Then*

1. $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$. *(De Morgan's Laws)*
2. $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$.
3. $\neg(\exists x_i\, \phi) \equiv \forall x_i\, \neg\phi$.
4. $\neg(\forall x_i\, \phi) \equiv \exists x_i\, \neg\phi$.
5. $\neg(\neg\phi) \equiv \phi$.

6. $\exists x_i\,(\phi \vee \psi) \equiv (\exists x_i\,\phi) \vee (\exists x_i\,\psi)$.

7. $\forall x_i\,(\phi \wedge \psi) \equiv (\forall x_i\,\phi) \wedge (\forall x_i\,\psi)$.

8. *If* x_i *does not appear free in* ψ, $\quad \forall x_i\,(\phi \vee \psi) \equiv (\forall x_i\,\phi) \vee \psi$.

9. *If* x_i *does not appear free in* ψ, $\quad \forall x_i\,(\phi \wedge \psi) \equiv (\forall x_i\,\phi) \wedge \psi$.

10. *If* x_j *does not appear in* ϕ, $\quad \forall x_i\,\phi \equiv \forall x_j\,\phi[x_i \leftarrow x_j]$.

Properties 8 through 10 still hold, if we replace $\forall$ by $\exists$, whereas for properties 6 and 7 this is not true in general.

A QBF is said to be in *prenex normal form*, if it is of the form

$$\phi = \exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n \;\; \psi,$$

where ψ is quantifier-free and $Q_n = \exists$ if n is odd and $Q_n = \forall$ otherwise. The string of quantifiers is called *prefix*, and ψ is called the *matrix* of the formula. The alternation of quantifiers is not always demanded in the definition of the normal form, however, it can be easily accomplished by introducing dummy variables that do not appear in the matrix.

Proposition 3. *Any QBF* ϕ *can be transformed to an equivalent one in prenex normal form.*

Proof sketch. First, assign different variable names to each quantification and each free variable. Then move the quantifiers out by the rules in Prop. 2. □

The decision problem of *quantified satisfiability* is now defined as follows:

$$\text{QSAT} = \big\{\langle\phi\rangle : \; \phi \text{ is a } \textbf{true} \text{ QBF in conjunctive prenex normal form}\big\}.$$

Note that, in particular, the formulas in QSAT are closed, since they evaluate to **true**.

2.2 PSPACE-Completeness

Theorem 4 (Stockmeyer/Meyer 1973)**.** QSAT *is* **PSPACE***-complete.*

Proof. In order to prove the theorem, we have to show that

1. QSAT $\in$ **PSPACE**.
2. For all $L \in$ **PSPACE** : $L \leq_{\log}$ QSAT.

Proof sketch of 1. Let $\phi = \exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n\, \psi(x_1, \ldots, x_n)$ be a QBF. Then ϕ can be evaluated as it is shown in Figure 1.

However, we cannot store the whole tree, since it has exponential size. But we can traverse it in a depth-first search manner like in the following recursive algorithm.

Algorithm. Truth(ϕ)

```
1: if φ is quantifier-free then
2:     return truth value of φ
3: end if

4: denote φ = Q_1 x_1 ... Q_n x_n ψ(x_1, ..., x_n)

5: b_0 ← Truth(Q_2 x_2 ... Q_n x_n ψ(false, x_2, ..., x_n))
6: b_1 ← Truth(Q_2 x_2 ... Q_n x_n ψ(true, x_2, ..., x_n))

7: if Q_1 = ∃ then
8:     return b_0 ∨ b_1
9: else
10:    return b_0 ∧ b_1
11: end if
```

If lines 5 and 6 are implemented with a stack that reuses space, then the stack size is bounded by the height of the tree which is linear in the input length.

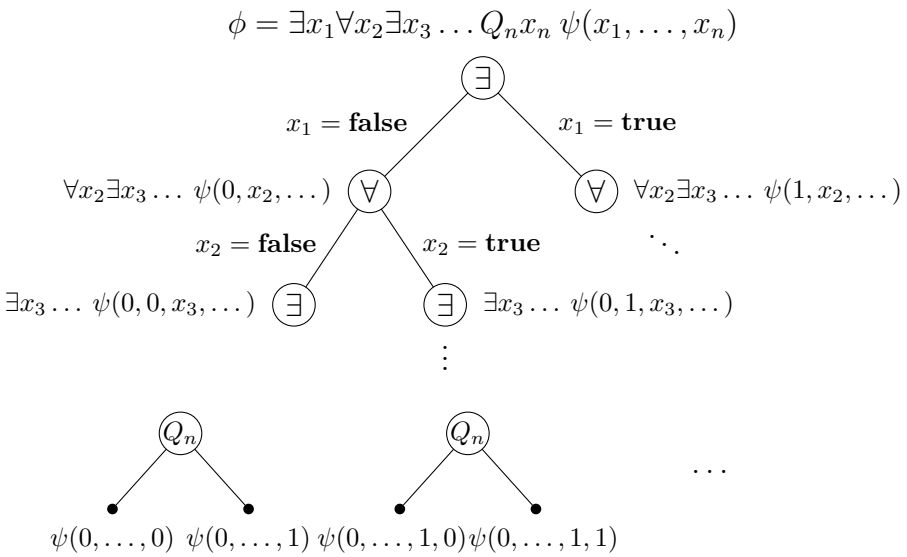

Figure 1: Evaluation tree.

Proof sketch of 2. Let $L \in \textbf{PSPACE}$. Then L is decidable by a Turing machine M that, for input x, uses polynomial space. Thus, the number of possible configurations is 2^m, where $m = \mathcal{O}(|x|^k)$, and we can encode each configuration as a bit vector X of length m. We consider the configuration graph $G(M, x)$ of M on input x, i. e. the graph that has as node set all possible configurations, and a directed edge between two nodes if and only if one configuration

yields the other in one step. Deciding now wether $x \in L$ is equivalent to the existence of a path in $G(M, x)$ from the initial configuration to an accepting configuration.

Therefore this proof makes use of the reachability method and is essentially a restatement of the proof of Savitch's Theorem in the language of logic. We want to construct a QBF $\psi_i(X, Y)$ that is **true** if and only if there is a path from the configuration encoded by the truth assignment for X to the configuration encoded by Y of length $\leq 2^i$. To complete the proof, we need to construct the formula $\psi_m(A, B)$, where A encodes the initial configuration and B the accepting configuration (we can assume w. l. o. g. that B is unique, since we can modify every program such that it clears its work tape and moves the cursor to the initial position before accepting). We use induction on i.

For $i = 0$, $\psi_0(A, B)$ should express that $A = B$ or that configuration B can be reached from A in one step. This formula can be written easily in DNF (why we use DNF instead of CNF will become clear in the induction step).

A first idea for the induction step might be, to define $\psi_{i+1}(A, B)$ as

$$\psi_{i+1}(A, B) = \exists Z \left[\psi_i(A, Z) \wedge \psi_i(Z, B)\right],$$

where Z encodes the unique midpoint of the path. However, this is a bad idea, since in each step the length of the formula at least doubles and we would end up with an exponentially large expression. Here we apply Savitch's trick of reusing space:

$$\psi_{i+1}(A, B) = \exists Z \forall X \forall Y \left[\big((X = A \wedge Y = Z) \vee (X = Z \wedge Y = B)\big) \Rightarrow \psi_i(X, Y)\right].$$

This means that we use the formula $\psi_i(X, Y)$ for both telling that there is a path from A to Z and from Z to B, using X and Y as placeholders.

Converting this formula to CNF could produce exponentially large expressions, but the DNF of this formula is small and easy to compute. However, the negation of a QBF in DNF is an expression in CNF by de Morgan's laws. So we reduced L to $\overline{\text{QSAT}}$, but this finishes the proof, since **PSPACE**, like any other deterministic complexity class, is closed under complement.

Thus, QSAT is **PSPACE**-complete. □

3 Shamir's Theorem

The complexity class **IP** consists of all languages L having an *interactive proof system*, i. e. a protocol between a *prover* Alice and a *verifier* Bob who *interact* on a common input x. The computational power of Alice is unbounded whereas Bob runs a *probabilistic*, *polynomial-time* algorithm. The correctness requirements for the proof are the following:

1. *Completeness:* If $x \in L$, then there exists a prover strategy such that Alice convinces Bob with probability at least $\frac{2}{3}$.

2. *Soundness:* If $x \notin L$, then, for any prover strategy, Alice convinces Bob with probability at most $\frac{1}{3}$.

Note that, since Bob can only process an polynomial amount of data, the number of rounds as well as the length of the messages exchanged in the protocol cannot be more then polynomial in the size of the input.

The following theorem characterizes the complexity class **IP** completely.

Theorem 5 (Shamir 1992)**.**

$$\mathbf{IP} = \mathbf{PSPACE}.$$

Proof.

$\subseteq$: Let $L \in \mathbf{IP}$. Then there exists an interactive proof system for L with a fixed verifier strategy. By traversing the tree of all possible interactions between Alice and Bob, we can compute an optimal prover strategy, i. e. a strategy that has the highest probability of convincing the verifier for every input x. This can be done in polynomial space in a way similar to part 1 of the proof of Theorem 4.

For input x we can now simulate the interaction between the optimal prover and the verifier, and enumerate over all possible coin tosses. We accept if and only if at least $\frac{2}{3}$ of the outcomes are accepting. This algorithm can be implemented in polynomial space as well and accepts if and only if $x \in L$.

$\supseteq$: It suffices to show that

$$\mathrm{QSAT} \in \mathbf{IP},$$

since **IP** is closed under reductions and QSAT is **PSPACE**-complete by Theorem 4.

Below we describe an interactive protocol that decides QSAT.

3.1 Arithmetization

First of all we convert a quantified Boolean expression ϕ into an arithmetic expression, the *arithmetization* A_ϕ of ϕ. This will allow us to use some results from number theory.

In order to arithmetize ϕ we replace its Boolean variables $x_i \in X$ by variables $z_i \in \mathbb{Z}$, where **true** corresponds to 1 and **false** to 0. The rules of conversion are now defined inductively as follows:

QBF ϕ	Arithmetization A_ϕ
$\neg x_i$	$1 - z_i$
$\psi_1 \vee \psi_2$	$A_{\psi_1} + A_{\psi_2}$
$\psi_1 \wedge \psi_2$	$A_{\psi_1} \cdot A_{\psi_2}$
$\exists x_i\, \psi$	$\sum_{z_i=0}^{1} A_\psi$
$\forall x_i\, \psi$	$\prod_{z_i=0}^{1} A_\psi$

The resulting expression is called Σ-Π *expression*.

Note that, for reasons that will become clear in the next lemma, the conversion of negation is only defined over variables and not over subexpressions. However, we can always assume ϕ to have this special form, since by de Morgan's laws we can push any negation sign all the way down to the variables.

Example. Consider the **true** QBF $\phi = \forall x_1 \left[\neg x_1 \vee \exists x_2 \forall x_3 \, (x_1 \wedge x_2) \vee x_3\right]$. Its arithmetization is

$$A_\phi = \prod_{z_1=0}^{1} \left[(1 - z_1) + \sum_{z_2=0}^{1} \prod_{z_3=0}^{1} (z_1 \cdot z_2 + z_3)\right].$$

Since ϕ is closed, A_ϕ evaluates to an integer, namely 2.

The following lemma shows, that positive arithmetizations always correspond to **true** QBFs.

Lemma 6. *Let ϕ be a closed QBF with negation only over variables. Then*

$$\phi \textit{ is } \mathbf{true} \iff A_\phi > 0.$$

Proof. Induction on the structure of ϕ. To use induction we have to allow free variables. So, in the case of free variables, we will prove the claim for any truth assignment.

First of all observe that arithmetizations are always nonnegative.

For $\phi = x_i$ we have $A_\phi = z_i$, therefore

$$\phi \text{ is } \mathbf{true} \iff x_i = \mathbf{true} \iff z_i = 1 \iff A_\phi > 0.$$

For $\phi = \neg x_i$ we have $A_\phi = 1 - z_i$, hence (that is why we allowed negation only over variables)

$$\phi \text{ is } \mathbf{true} \iff x_i = \mathbf{false} \iff z_i = 0 \iff A_\phi > 0.$$

Let ϕ be of the form $\phi = \psi_1 \vee \psi_2$. Then $A_\phi = A_{\psi_1} + A_{\psi_2}$ and it follows by induction that

$$\begin{aligned} \phi \text{ is } \mathbf{true} &\iff \psi_1 \text{ is } \mathbf{true} \quad \text{or} \quad \psi_2 \text{ is } \mathbf{true} \\ &\iff A_{\psi_1} > 0 \quad \text{or} \quad A_{\psi_2} > 0 \\ &\iff A_\phi > 0. \end{aligned}$$

Let ϕ be of the form $\phi = \exists x_i \, \psi$. Then $A_\phi = \sum_{z_i=0}^{1} A_\psi$ and it follows by induction that

$$\begin{aligned} \phi \text{ is } \mathbf{true} &\iff \psi(x_i = \mathbf{true}) \text{ is } \mathbf{true} \quad \text{or} \quad \psi(x_i = \mathbf{false}) \text{ is } \mathbf{true} \\ &\iff A_\psi(z_i = 1) > 0 \quad \text{or} \quad A_\psi(z_i = 0) > 0 \\ &\iff A_\phi > 0. \end{aligned}$$

The remaining cases can be proven similarly. □

Thus, instead of convincing Bob that ϕ is true, Alice can also convince Bob that $A_\phi > 0$. To do so, Alice may wish to send the value of A_ϕ to Bob. However, this is not always possible as the following example shows.

Example. Consider the QBF $\phi = \forall x_1 \forall x_2 \cdots \forall x_{k-1} \exists x_k \, (x_k \vee \neg x_k)$. Then

$$A_\phi = \prod_{z_1=0}^{1} \prod_{z_2=0}^{1} \cdots \prod_{z_{k-1}=0}^{1} \sum_{z_k=0}^{1} \big[z_k + (1 - z_k)\big] = 2^{2^{k-1}},$$

since the sum evaluates to 2 and each of the $k-1$ products squares the previous value. This number has exponentially many bits and cannot be transmitted.

One idea to solve this problem is to do all computations modulo an integer of polynomial size.

3.2 Reduction to a Finite Field

Modular arithmetic solves the problem of exponential values of arithmetizations. However, we have to be careful not to choose a divisor of such values as modulus, because this would destroy the useful characterization in Lemma 6.

On the other hand, it will turn out to be important for the verifier Bob to know that the computations are done in a field. Otherwise, Alice would be able to cheat. Therefore we wish to use a small prime p as modulus, what means arithmetic in the finite field $\mathbb{F}_p$.

The following proposition reaches both goals.

Proposition 7. *For every* Σ-Π *expression* $A \neq 0$ *of length* n*, there exists a prime* $p \in [2^n, 2^{3n}]$ *such that*

$$A \neq 0 \pmod{p}.$$

For the proof, we need an upper bound for the value of an arithmetization and a lower bound for the number of primes in the interval.

Lemma 8. *Let* A_ϕ *be a* Σ-Π *expression of length* n*. Then*

$$A_\phi \leq 2^{2^n}.$$

Proof. Induction on the structure of ϕ (again, not necessarily closed, compare with the proof of Lemma 6).

For literals $\phi = (\neg)x_i$, it holds that $A_\phi \leq 1 \leq 2^{2^1}$.

Let ϕ be of the form $\phi = \psi_1 \circ \psi_2$, where $\circ \in \{\vee, \wedge\}$. By induction, we have $A_{\psi_1} \leq 2^{2^\ell}$ and $A_{\psi_2} \leq 2^{2^m}$ with $\ell + m \leq n$. Hence, the value of A_ϕ can be at most (in the case of a conjunction)

$$A_\phi \leq A_{\psi_1} \cdot A_{\psi_2} \leq 2^{2^\ell} \cdot 2^{2^m} = 2^{2^\ell + 2^m} \leq 2^{2^n}.$$

Finally, let ϕ be of the form $\phi = Qx_i\,\psi$, where $Q \in \{\exists, \forall\}$. Then $A_\psi \leq 2^{2^m}$ with $m < n$. So, the value of A_ϕ can be at most (in the case of a universal quantifier)

$$A_\phi \leq \prod_{z_i=0}^{1} A_\psi \leq 2^{2^m} \cdot 2^{2^m} = 2^{2\cdot 2^m} = 2^{2^{m+1}} \leq 2^{2^n}. \qquad \square$$

Lemma 9. *For $n \geq 3$,*

$$\sqrt{n} \leq \pi(n) \leq n,$$

where $\pi(n)$ denotes the number of primes up to n.

Proof. The upper bound is trivial.

For the lower bound consider the integers $2, \ldots, n$. We now apply the Sieve of Eratosthenes. This means, for every prime $p \leq \sqrt{n}$ we strike all multiples of p off the list. After each step at least $(p-1)/p$ of the previous numbers will remain in the list. In the end, the list will contain exactly the prime numbers up to n.

Therefore we can compute directly

$$\pi(n) \geq n \prod_{p \leq \sqrt{n}} \frac{p-1}{p} \geq n \prod_{i=2}^{\lfloor \sqrt{n} \rfloor} \frac{i-1}{i} = n/\lfloor \sqrt{n} \rfloor \geq \sqrt{n},$$

where the first product ranges over all primes up to $\sqrt{n}$. $\square$

Theorem 10 (Chinese Remainder Theorem)**.** *Let $a_1, \ldots, a_k \in \mathbb{Z}$, and let $p_1, \ldots, p_k \in \mathbb{N}_{>0}$ be pairwise coprime.*

Then the system of simultaneous congruences

$$\begin{aligned} x &= a_1 \pmod{p_1} \\ &\vdots \\ x &= a_k \pmod{p_k} \end{aligned}$$

has a unique solution x modulo $\prod_{i=1}^{k} p_i$.

Proof of Prop. 7. Denote by $p_1, \ldots, p_k$ all primes in the interval $[2^n, 2^{3n}]$. Using Lemma 9 we obtain

$$k = \pi\big(2^{3n}\big) - \pi\big(2^n\big) \geq \sqrt{2^{3n}} - 2^n > 2^n.$$

Suppose now for the sake of a contradiction that

$$A = 0 \pmod{p_i} \qquad \forall i \in \{1, \ldots, k\}.$$

Then, by the Chinese Remainder Theorem, $A = 0$ modulo $\prod_{i=1}^{k} p_i > 2^{2^n}$. But since $A \leq 2^{2^n}$ (Lemma 8), we conclude $A = 0$ in $\mathbb{Z}$. Contradiction. $\square$

3.3 Polynomials and Simple Expressions

In order to convince Bob of the truth of ϕ, Alice has to reduce the size of the expression A_ϕ gradually. This is done by the following construction.

Definition 11. Let A be a Σ-Π expression, where the leftmost symbol is $\sum_{z_i=0}^{1}$ or $\prod_{z_i=0}^{1}$. The *functional form* A' is defined by eliminating the leftmost $\sum_{z_i=0}^{1}$ or $\prod_{z_i=0}^{1}$ symbol in A, and can be considered as a polynomial

$$q(z_i) \in \mathbb{Z}[z_i].$$

The *randomized form* of A is $A'(z_i = r)$, where $r \in_R \mathbb{F}_p$ is a random number supplied by the verifier Bob.

The coefficients of the polynomial $q(z_i)$ are hard to compute in general. However, for the prover Alice it is possible. Alice may wish to transmit those coefficients, and here the next problem arises. Indeed, the coefficients are small enough since we use arithmetic in $\mathbb{F}_p$, but the number of coefficients can be far too large, as the following example shows.

Example. Consider the QBF $\phi = \forall x_1 \forall x_2 \ldots \forall x_k \, (x_1 \vee x_2 \vee \cdots \vee x_k)$. Then

$$A'_\phi = \prod_{z_2=0}^{1} \cdots \prod_{z_k=0}^{1} (z_1 + z_2 + \cdots + z_k) = \prod_{i=1}^{2^{k-1}} (z_1 - c_i) \quad \text{with } 0 \le c_i \le k.$$

Therefore $q(z_1)$ is a dense polynomial of exponentially high degree $\deg q(z_i) = 2^{k-1}$.

The example shows that a large number of products can be responsible for the explosion of the degree. Therefore we should try to eliminate as much universal quantifiers from our QBFs as possible.

Definition 12. A QBF ϕ is called *simple*, if any occurrence of a variable is separated by at most one universal quantifier from its point of quantification.

Example. The QBF

$$\phi = \forall x_1 \forall x_2 \exists x_3 \left[(x_1 \vee x_2) \wedge \forall x_4 \, (x_2 \vee x_3 \vee x_4) \right]$$

is simple, since

1. the single occurrence of x_1 is separated only by $\forall x_2$,
2. the first occurrence of x_2 is not separated by any universal quantifier, and the second occurrence is separated only by $\forall x_4$,
3. the single occurrence of x_3 is separated only by $\forall x_4$, and
4. the single occurrence of x_4 is not separated by any universal quantifier

from its point of quantification. On the other hand, the QBF

$$\psi = \forall x_1 \forall x_2 \left[(x_1 \vee x_2) \wedge \forall x_3 \, (\neg x_1 \vee x_3) \right]$$

is not simple, since the second occurrence of x_1 is separated from its point of quantification by both $\forall x_2$ and $\forall x_3$.

For simple QBFs, the degree of the polynomial is linear in the size of the formula.

Lemma 13. *Let ϕ be a simple QBF of length n, and let $q(z_i)$ be the polynomial of the functional form of A_ϕ. Then*

$$\deg q(z_i) \leq 2n.$$

Proof. For quantifier-free subexpressions, you can show by straightforward induction that the degree of the resulting polynomial in z_i is bounded above by the size of the subexpression. Summations can only change the coefficients, but not the degree, and each product can at most double the degree. But since ϕ is simple, such a doubling can happen only once. □

The following lemma shows that we can assume simple QBFs without loss of generality.

Lemma 14. *Any QBF ϕ of length n can be transformed in logarithmic space to an equivalent simple expression.*

Proof. Let ϕ be of the form $\phi = \ldots Q x_i \ldots \forall x_j \, \psi(x_i)$, where $Q \in \{\exists, \forall\}$ and $\forall x_j$ is the first universal quantifier after $Q x_i$. We transform ϕ as follows:

$$\begin{aligned} \phi' &= \ldots Q x_i \ldots \forall x_j \exists x_{i'} \, (x_i \Leftrightarrow x_{i'}) \wedge \psi(x_{i'}) \\ &= \ldots Q x_i \ldots \forall x_j \exists x_{i'} \left[(x_i \wedge x_{i'}) \vee (\neg x_i \wedge \neg x_{i'}) \right] \wedge \psi(x_{i'}). \end{aligned}$$

That means, we give all occurrences of x_i after $\forall x_j$ a new name $x_{i'}$, and we express the equivalence of x_i and $x_{i'}$ by a small formula without introducing new universal quantifiers. If we iterate this procedure from the left to the right for every variable and every universal quantifier behind it, then, after $\mathcal{O}(n^2)$ steps, we end up with a simple expression that is equivalent to the original one.

It should be clear that this transformation can be done in logarithmic space, since only local changes are made. □

3.4 The Interactive Protocol

Now we are able to give a description of the interactive protocol.

So let ϕ be a simple QBF. Alice wants to give a proof that $A_\phi \neq 0$. In the setup of the protocol she chooses a prime $p \in [2^n, 2^{3n}]$ according to Prop. 7 and computes the value of the arithmetization $a \leftarrow A_\phi \pmod p$ (see Fig. 2).

Prover	Verifier
Choose $p \in [2^n, 2^{3n}]$. Compute $a \leftarrow A_\phi \pmod p$.	
$\xrightarrow{p,\ a}$	
	Verify $a \neq 0$, $p \in [2^n, 2^{3n}]$, and $p \in$ PRIMES.

Figure 2: Protocol setup.

Then she sends both p and a to Bob who checks that $a \neq 0$ and $p \in [2^n, 2^{3n}]$. Moreover, Bob tests the primality of p, which can be done in polynomial time with the AKS-algorithm (see [AKS04]). If any of the tests fails he rejects.

At any intermediate step of the protocol, the current expression A is split up into

$$A = A_1 + A_2 \qquad \text{or} \qquad A = A_1 \cdot A_2,$$

depending on the structure of A, such that A_2 starts with the leftmost $\sum_{z_i=0}^{1}$ or $\prod_{z_i=0}^{1}$ symbol of A. Since A_1 contains no such symbols, Bob can compute the value $a_1 \leftarrow A_1$ himself. Then Alice and Bob repeatedly execute the following simplification step (see Fig. 3):

1. Bob sets $A \leftarrow A_2$, and $a \leftarrow a - a_1 \pmod p$ or $a \leftarrow a/a_1 \pmod p$ (depending on the decomposition of A).

2. Alice computes the polynomial $q(z_i)$ of A' and sends the coefficients to Bob.

3. Bob verifies $a = q(0) + q(1) \pmod p$ or $a = q(0) \cdot q(1) \pmod p$ (depending on the first symbol of A_2). Otherwise, he rejects.

4. Bob chooses $r \in_R \mathbb{F}_p$ at random and sends it to Alice. Then he sets $A \leftarrow A'(z_i = r) \pmod p$ and $a \leftarrow q(r) \pmod p$.

The procedure stops if A_2 is free of $\sum_{z_i=0}^{1}$ and $\prod_{z_i=0}^{1}$ symbols, and Bob accepts if and only if $a = a_1$.

Example. Consider the simple QBF $\phi = \forall x_1 \left[\neg x_1 \vee \exists x_2 \forall x_3 \, (x_1 \wedge x_2) \vee x_3\right]$. For simplicity, we omit modular reductions in this example.

In the setup,

$$A \leftarrow A_\phi = \prod_{z_1=0}^{1} \left[(1 - z_1) + \sum_{z_2=0}^{1} \prod_{z_3=0}^{1} (z_1 \cdot z_2 + z_3) \right],$$

and Alice computes the value $a \leftarrow 2$ of A.

Prover		Verifier
		$A \leftarrow A_2$. $a \leftarrow a - a_1 \pmod p$, or $a \leftarrow a/a_1 \pmod p$.
Compute $q(z_i)$ of A'.		
	$\xrightarrow{q(z_i)}$	
		Verify $a = q(0)+q(1) \pmod p$, or $a = q(0) \cdot q(1) \pmod p$. Choose $r \in_R \mathbb{F}_p$.
	$\xleftarrow{r}$	
		$A \leftarrow A'(z_i = r) \pmod p$. $a \leftarrow q(r) \pmod p$.

Figure 3: Simplification step.

In the first round, A already starts with a $\prod_{z_1=0}^{1}$ symbol, so Alice computes the polynomial $q(z_1) = z_1^2 + 1$ of

$$A' = (1 - z_1) + \sum_{z_2=0}^{1} \prod_{z_3=0}^{1} (z_1 \cdot z_2 + z_3).$$

Bob checks that $a = 2 = 2 \cdot 1 = q(0) \cdot q(1)$ and chooses a number at random, say 3. Then he sets

$$A \leftarrow A'(z_1 = 3) = (1 - 3) + \sum_{z_2=0}^{1} \prod_{z_3=0}^{1} (3z_2 + z_3)$$

and $a \leftarrow q(3) = 10$.

In the second round, Bob sets

$$A \leftarrow A_2 = \sum_{z_2=0}^{1} \prod_{z_3=0}^{1} (3z_2 + z_3)$$

and $a \leftarrow a - a_1 = 10 - (-2) = 12$. Alice computes the polynomial $q(z_2) = 9z_2^2 + 3z_2$ of

$$A' = \prod_{z_3=0}^{1} (3 \cdot z_2 + z_3).$$

Bob checks that $a = 12 = 0 + 12 = q(0) + q(1)$ and chooses a number at random, say 2. Then he sets

$$A \leftarrow A'(z_2 = 2) = \prod_{z_3=0}^{1} (z_3 + 6)$$

and $a \leftarrow q(2) = 9 \cdot 4 + 3 \cdot 2 = 42$.

In the third round, Alice computes the polynomial $q(z_3) = z_3 + 6$ of

$$A' = z_3 + 6.$$

Bob checks that $a = 42 = 6 \cdot 7 = q(0) \cdot q(1)$ and chooses a number at random, say 5. Then he sets $A \leftarrow A'(z_3 = 5) = 5 + 6 = 11$ and $a \leftarrow q(5) = 5 + 6 = 11$. Since $a_1 = 11 = a$, he finally accepts.

To complete the proof of the main theorem, it remains to show the correctness of the protocol.

Theorem 15.

1. *When ϕ is true and Alice is honest, Bob will always accept the proof.*
2. *When ϕ is false, Bob accepts the proof with negligible probability.*

Proof. The completeness of the protocol is clear from the construction, because Alice is able to provide all the polynomials and Bob will always accept.

For the soundness, assume that $A_\phi = 0$ and still Alice starts the protocol with a value $a' \neq 0$. Then she has to provide a wrong polynomial $q'(z_i)$ in the i-th round, because $q'(0) + q'(1)$ (or $q'(0) \cdot q'(1)$) must yield the wrong value. Since $0 \neq q(z_i) - q'(z_i)$ is a polynomial of degree at most $2n$ by Lemma 13, it has at most $2n$ roots in $\mathbb{F}_p$ (here it becomes important that the computations are done in a field). Therefore, the probability that $q'(z_i)$ yields the correct value when evaluated at a random $r \in_R \mathbb{F}_p$ is at most

$$\Pr[\text{error in the } i\text{-th round}] \leq \frac{2n}{p} \leq \frac{2n}{2^n},$$

by the choice of p. Since the random numbers are chosen by Bob independently, after the whole $m \leq n$ rounds the probability of a false positive is

$$\begin{aligned} \Pr[\text{error}] &= 1 - \Pr[\text{no error}] \\ &= 1 - \prod_{i=1}^{m} \Pr[\text{no error in the } i\text{-th round}] \\ &\leq 1 - \left(1 - \frac{2n}{2^n}\right)^n, \end{aligned}$$

which gets arbitrarily small as $n \to \infty$. □

Remark.

1. The interactive protocol has perfect soundness, and since the random numbers can be made public by Bob, it is actually of the Arthur-Merlin type.
2. The prover strategy can be implemented in polynomial space.

References

[AKS04] M. Agrawal, N. Kayal, N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160 (2), pp. 781-793, 2004.

[Ba90] L. Babai. E-mail and the unexpected power of interaction. *Structure in Complexity Theory Conf.*, pp. 30-44, 1990.

[GMR85] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof-systems. *Proc. 17th ACM Symp. on the Theory of Computing*, pp. 291-304, 1985.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39 (4), pp. 859-868, 1992.

[Pa83] C. H. Papadimitriou. Games against nature. *Proc. 24th IEEE Symp. on the Foundations of Computer Science*, pp. 446-450, 1983.

[Pa94] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, 1994.

[SchPr98] U. Schöning and R. Pruim. *Gems of Theoretical Computer Science*. Springer, 1998.

[Sh92] Adi Shamir. IP=PSPACE. *Journal of the ACM*, 39 (4), pp. 869-877, 1992.

[StMe73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. *Proc. 5th ACM Symp. on the Theory of Computing*, pp. 1-9, 1973.

[To89] S. Toda. On the computational power of PP and $\oplus$P. *Proc. of the 30th IEEE Symp. on Foundations of Computer Science*, pp. 514-519, 1989.

[Va79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8, pp. 189-201, 1979.