IP = PSPACE

ADI SHAMIR

The Weizmann Institute of Science, Rehovot, Israel

Abstract. In this paper, it is proven 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.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*bounded-action devices* (*e.g., Turing machines, random access machines*); F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*interactive computation, probabilistic computation, relations among modes*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*complexity hierarchies, relations among complexity classes*

General Terms: Algorithms, Theorem

Additional Key Words and Phrases: Interactive proofs, IP, PSPACE

1. *Introduction*

The class IP of languages that have efficient interactive proofs of membership was introduced by Goldwasser et al. [8], and in a slightly different form by Babai [1] (the equivalence between these models was established by Goldwasser and Sipser [7]). A language L belongs to IP if a probabilistic polynomial time verifier V can be convinced by some prover P to accept any $x \in L$ with overwhelming probability, but cannot be convinced by any prover P' to accept any $x \notin L$ with a nonnegligible probability. Goldreich et al. [6] showed that IP contains some languages believed not to be in NP, but its exact characterization remained a major open problem for several years. In a breakthrough paper, Lund et al. [10] made ingenious use of earlier results by Valiant [12], Toda [11] Beaver and Feigenbaum [2] and Lipton [9] to prove that IP contains the polynomial hierarchy PH. In this paper, we use surprisingly simple techniques to extend the previous results and prove that IP contains PSPACE. Since every IP language is trivially accepted by the PSPACE machine that traverses the tree of all the possible interactions, this result completely characterizes IP.

The interactive proofs introduced in this paper use only public coins, are accepted with probability 1 when the prover is honest, and require only logarithmic workspace when the verifier is given a 2-way access to his random

Author's address: Applied Mathematics Department, The Weizmann Institute of Science, Rehovot, Israel.

tape. They can be turned into zero knowledge proofs under the sole assumption that one-way functions exist by using known techniques.

2. *Quantified Boolean Formulas*

The class of quantified Boolean formulas (QBF) is defined as the closure of the set of Boolean variables x_i and their negations $\bar{x}_i$ under the operations $\wedge$ (and), $\vee$ (or), $\forall x_i$ (universal quantification), and $\exists x_i$ (existential quantification). These operations can be mixed in any order, and in particular we do not require that all the quantifiers appear in a leftmost prefix. For the sake of convenience, we add the boolean equality operation, denote vectors of boolean variables by X_i, and extend the operations $=, \forall, \exists$ from single variables x_i to vectors of variables X_i in the obvious way.

A QBF in which all the variables are quantified is called *closed*, and 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 $\{T, F\}$.

The interactive proofs presented in this paper can only handle QBFs of a special type:

Definition. 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, since:

(1) x_1 is used once, and only $\forall x_2$ separates its point of quantification from its point of use,
(2) x_2 is used twice; its first use is not separated from its point of quantification by any universal quantifiers, whereas its second use is separated only by $\forall x_4$,
(3) x_3 is used once, and separated only by $\forall x_4$,
(4) x_4 is used once, and not separated by any universal quantifiers.

Example. $\forall x_1 \forall x_2 [(x_1 \wedge x_2) \wedge \forall x_3 (\bar{x}_1 \wedge x_3)]$ is not simple, since the second occurrence of x_1 (which is negated) is separated from its point of quantification by both $\forall x_2$ and $\forall x_3$.

The notion of simple QBFs seems to be quite restrictive. However, we can prove:

THEOREM 1. *Every QBF of size n can be transformed into an equivalent simple QBF whose size is polynomial in n.*

PROOF. To turn a given QBF into a simple QBF, we completely rename all the Boolean variables after each universal quantifier, by introducing new existentially quantified Boolean variables x_i^j to denote the jth name of the ith variable, and equating each x_i^j with its previous name x_i^{j-1}. Consider, for example, the QBF

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \cdots Q(x_1, x_2, \ldots).$$

in which Q is quantifier free. The transformed QBF is:

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

where $x_i^{l_i}$ 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). □

3. *The Arithmetization of QBF*

The problem of deciding whether a given closed QBF is true or false can be expressed in an arithmetic form by using the following syntactic transformations:

(1) Replace each Boolean variable x_i by a new variable z_i which can range over the (positive and negative) integers $\mathbb{Z}$.
(2) Replace each occurrence of $\bar{x}_i$ by $(1 - z_i)$.
(3) Replace $\wedge$ by integer multiplication $\cdot$, $\vee$ by integer addition $+$, the universal quantification $\forall x_i$ by the integer product $\prod_{z_i \in \{0,1\}}$, and the existential quantifier $\exists x_i$ by the integer sum $\sum_{z_i \in \{0,1\}}$.

Example. Consider the true QBF:

$$B = \forall x_1 \exists x_2 [(x_1 \wedge x_2) \vee \exists x_3 (\bar{x}_2 \wedge x_3)].$$

Its arithmetization yields:

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

which can be evaluated to the integer 2.

THEOREM 2. *A closed QBF B is true iff the value of its arithmetic form A is nonzero.*

PROOF. By straightforward induction on the structure of B. □

Remarks.

(1) A problem may arise if we try to express a negated $\bar{B}$ directly as $(c - A)$, since the value c of the arithmetic form A of B is difficult to compute. We avoid this difficulty by allowing negated variables but no negated expressions in our definition of QBFs. This is not a real limitation, since we can always push the negations in B all the way to the variables without increasing its structural complexity.
(2) We cannot replace the boolean $=$ by arithmetic $=$, since we want to interpret different non-zero integers as the same Boolean value T. We thus have to replace $x_i = x_j$ by $(x_i \wedge x_j) \vee (\bar{x}_i \wedge \bar{x}_j)$ and arithmetize the expression as $z_i z_j + (1 - z_i)(1 - z_j)$.

When B is true, the value of A can be quite large. However, we can upper bound this value as follows:

THEOREM 3. *Let B be a closed QBF of size n. Then the value of its arithmetic form A cannot exceed* $O(2^{2^n})$.

PROOF. For any (potentially open) subexpression B' of B, define $v(B')$ as the maximal value of the arithmetic form of B' under all the possible $0/1$ substitutions to the free variables.

This value satisfies:

(1) If B' is x_t or $\bar{x}_t$, then $v(B') = 1$.
(2) If B' is $B'' \vee B'''$, then $v(B') \leq v(B'') + v(B''')$.
(3) If B' is $B'' \wedge B'''$, then $v(B') \leq v(B'') \cdot v(B''')$.
(4) If B' is $\exists x_t B''$, then $v(B') \leq 2v(B'')$.
(5) If B' is $\forall x_t B''$, then $v(B') \leq v(B'')^2$.
(6) When B' is closed, $v(B')$ coincides with the value of its arithmetic form.

It is easy to verify that $O(2^{2^n})$ satisfies all the recursive inequalities. This bound cannot be substantially improved, since for

$$B = \forall x_1 \forall x_2 \cdots \forall x_{n-1} \exists x_n (x_n \vee \bar{x}_n),$$

$v(\exists x_n (x_n \vee \bar{x}_n)) = 2$, and each universal quantifier squares the previous value. □

Since such large numbers cannot be handled by the polynomial time verifiers in our interactive protocols, we reduce them modulo some smaller prime p.

THEOREM 4. *Let B be a closed QBF of size n. Then there exists a prime p of length polynomial in n such that $A \neq 0 \pmod p$ iff B is true.*

PROOF. Assume first that A is a non-zero integer. If it is zero modulo all the polynomial size primes, then by the Chinese remainder theorem it is zero modulo their product as well. Since by the prime number theorem this product is $\Omega(2^{2^{n^d}})$ for any desired constant d, this contradicts the assumption that A is non-zero and at most $O(2^{2^n})$. On the other hand, if B is false, then $A = 0$ modulo any prime. □

4. *Interactive Proofs for PSPACE Languages*

Since deciding the truth of (simple) QBFs is known to be PSPACE complete (see Garey and Johnson [5]), it suffices to demonstrate the existence of interactive proofs for this problem. 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. This prime can be chosen by the prover P and sent to the verifier V (along with a written proof of primality) as the first step of the interactive proof, since even an infinitely powerful cheating prover cannot find such a prime if $A = 0$. Alternatively, we can ask V to choose a random prime p, and use the fact that for most p, $A \neq 0 \pmod p$ iff B is true. This makes it possible to prove both membership and non-membership by the same protocol, but has imperfect completeness (even an honest prover may be unable to prove a correct statement if the verifier chooses a prime divisor of A).

Given a closed arithmetic expression A, we define its *functional form* A' by eliminating the leftmost $\sum_{z_t \in \{0,1\}}$ or $\prod_{z_t \in \{0,1\}}$ symbol, and considering A' as a polynomial function $q(z_t)$ of one free variable z_t. The *randomized form* of A is $A'(z_t = r)$ in which z_t is set to a random number r modulo p supplied by the verifier. This randomized form can again be evaluated to a constant, but its number of $\sum$ and $\prod$ symbols is reduced by 1. Note that the simplicity of A is preserved by these transformations.

Example. Consider the true QBF:

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

and assume that p is large enough so that the following calculations are not affected by modular reductions. The arithmetic form of B is

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

whose value is 2. The functional form of A is:

$$A' = \left[(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2 + z_3) \right].$$

By evaluating all the summations and products, the infinitely powerful prover can express A' as the polynomial

$$q(z_1) = z_1^2 + 1.$$

The randomized form of A' with $z_1 = 3$ can be expressed as

$$A'(z_1 = 3) = \left[(1 - 3) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (3z_2 + z_3) \right],$$

whose value 10 can be deduced from $q(3) = 3^2 + 1 = 10$.

The polynomial $q(z_i)$ that represents A' can have an exponentially high degree. Consider, for example, the QBF

$$B = \forall x_1 \forall x_2 \cdots \forall x_n (x_1 \vee x_2 \vee \cdots \vee x_n),$$

whose arithmetization is

$$A = \prod_{z_1 \in \{0,1\}} \prod_{z_2 \in \{0,1\}} \cdots \prod_{z_n \in \{0,1\}} (z_1 + z_2 + \cdots + z_n).$$

The polynomial $q(z_1)$ of its functional form A' is the product of 2^{n-1} terms of the form $(z_1 + c)$ for $0 \le c \le n$, which is a dense polynomial of degree 2^{n-1}. Such polynomials cannot be handled by the polynomial time verifier during the interactive proof. However, for simple QBFs we can prove:

THEOREM 5. *If B is simple, then the degree of the polynomial $q(z_t)$ that describes the functional form of A grows at most linearly with the size of B.*

PROOF. Let x_t be the leftmost quantified variable in B. The degree of z_t created by any quantifier-free subexpression of B is bounded by the size of the subexpression. Summations over arbitrarily many other variables can change the coefficients but not the degree of the polynomial, and each product can at most double the degree. The result follows from the fact that in simple B such a doubling can occur at most once. □

Remark. By adding sufficiently many dummy variables to the quantifier-free subexpressions of B, this degree can be reduced to 3, and thus $q(z_t)$ can be represented by just four numbers in $\mathbb{Z}_p$. Consider for example $B = \forall x_t B'$ where x_t occurs k times in B'. This expression can be replaced by

$$\forall x_t^1 \exists x_t^2 \cdots \exists x_t^k \left(x_t^1 = x_t^2 = \cdots = x_t^k \right) \wedge B'',$$

where B'' is obtained from B' by using a different x_t^j variable in each occurrence of x_t in B'. If B is simple and the multiple equality is arithmetized as $z_t^1 \cdot z_t^2 \cdots z_t^k + (1 - z_t^1)(1 - z_t^2) \cdots (1 - z_t^k)$, the degree of each functional form is at most 3.

The interactive protocol for proving that $A \neq 0 \pmod p$ is very simple. The prover P sends the claimed value a of $A \pmod p$ to the verifier 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 Σ or $\prod$ symbol of A. P and V then repeatedly execute the following simplification steps:

(1) If A_2 is empty, V stops and accepts the claim iff $a = a_1$.
(2) 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 $a = 0 \pmod p$.
(3) Otherwise, P sends the polynomial descriptions $q(z_t)$ of A' to V. V checks that $a = q(0) + q(1) \pmod p$ or $a = q(0) \cdot q(1) \pmod p$ (depending on the first symbol of A_2), sends a random $r \in \mathbb{Z}_p$ to P, replaces A by $A'(z_t = r) \pmod p$, and replaces a by $q(r) \pmod p$.

Example. Consider once more the expression A of the previous example, whose claimed value is $a = 2$. When P sends its polynomial $q(z_1) = z_1^2 + 1$ to V, V checks that $q(0) \cdot q(1) = 1 \cdot 2 = a$, sends $z_1 = 3$ to P, replaces A by $A'(z_1 = 3)$, and replaces a by $q(3) = 10$. The new A starts with a nonempty A_1, whose value $a_1 = (1 - 3) = -2$ can be computed by V. A and a are now adjusted to

$$A = \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (3z_2 + z_3),$$

$$a = 10 - (-2) = 12.$$

The functional form of this A is

$$A' = \prod_{z_3 \in \{0,1\}} (3z_2 + z_3),$$

whose polynomial representation is $q(z_2) = 9z_2^2 + 3z_2$. When V gets this polynomial from P, he checks that $q(0) + q(1) = 0 + 12 = 12 = a$, and picks $z_2 = 2$ as his random choice. A and a are again adjusted to

$$A = \prod_{z_3 \in \{0,1\}} (6 + z_3),$$

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

P now sends $q(z_3) = z_3 + 6$ as the polynomial representation of A', and V checks that $q(0) \cdot q(1) = 6 \cdot 7 = 42 = a$. Finally, V chooses $z_3 = 5$, and verifies by himself that the value of

$$A'(z_3 = 5) = (6 + 5) = 11$$

is the same as $a = q(5) = 5 + 6 = 11$.

THEOREM 6

(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) The proof of this part is similar to the one used by Lund et al. [10] in their permanent proving protocol. A cheating prover who supplied an incorrect value of a must provide an incorrect polynomial $q(z_1)$ to support his claim, since a is checked against $q(0) + q(1) \pmod p$ or $q(0) \cdot q(1) \pmod p$. 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 $\mathbb{Z}_p$. 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 subexpressions, until he is exposed with overwhelming probability when V evaluates the final subexpression by himself. □

Remark. 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, as shown by Boppana et al. [3].

5. *Space-Bounded Verifiers*

The interactive proofs introduced in Section 4 require polynomial time and polynomial space verifiers. In this section, we prove that the space bound can be greatly improved.

Definition. A verifier is called *weak* if

(1) its running time is polynomial,
(2) its workspace is logarithmic,
(3) it has a two-way read-only access to a random tape,
(4) its messages consist solely of the random bits it reads.

THEOREM 7. *Any PSPACE language can be accepted by a weak verifier.*

Remarks. Since the accessible portion of the random tape is polynomial, we cannot "cheat" by using the head position or the location of desirable substrings as super-logarithmic auxiliary storage. Note that without the two-way access, Fortnow and Sipser [4], proved that only languages in P can be accepted by such verifiers, and without condition 4, Condon and Rompel [private communication] have already shown that logspace verifiers can accept all of IP.

PROOF (SKETCH). To use such weak verifiers in our interactive proofs, we consider the particular PSPACE complete class of QBFs obtained by the standard reduction from deterministic polynomial space Turing machine com-

putations. We encode each configuration of such a computation (which consists of the contents of the tape, the position of the head, and the internal state) as a polynomially long vector X of Boolean variables. If X_1 and X_2 are two configurations, then we can recursively express the existence of a legal transition of length $2^k (k \geq 1)$ between them by:

$$Q(X_1, X_2, 2^k) = \exists X_3 \forall x_4 \exists X_5 \exists X_6 \{[x_4 \Rightarrow (X_5 = X_1) \wedge (X_6 = X_3)] \wedge [\bar{x}_4 \Rightarrow (X_5 = X_3) \wedge (X_6 = X_2)] \wedge Q(X_5, X_6, 2^{k-1})\}.$$

The final $Q(X_t, X_{t+1}, 1)$ is the quantifier free 3CNF formula that characterizes a single move of the given Turing machine. In this expression X_3 represents the middle configuration of the computation, the single Boolean variable x_4 chooses which half of the computation we consider, X_5 and X_6 are new configuration names, and the rest of the expression states that both halves are legal transitions of length 2^{k-1}. When X_1 is the initial configuration and X_2 is the unique accepting configuration (with tape, initial head position, and accepting state), we can express the acceptance condition by the polynomially long QBF obtained by unrolling this recursive definition.

Such QBFs are simple by definition. Their innermost 3CNF formulas can be arithmetized with de Morgan's laws (replacing each clause such as $z_t + z_j + z_k$ by the logically equivalent $1 - (1 - z_t)(1 - z_j)(1 - z_k)$) to yield only 0/1 values. Such values are preserved by the products that result from universal quantifiers. Since the Turing machine computation is deterministic, the existentially quantified configurations X_3, X_5, and X_6 are uniquely determined by the endpoint configurations X_1, X_2, and the selector x_4. When such a $\exists X_t$ is arithmetized, at most one of the exponentially many summands can be non-zero. We can thus prove by induction that the arithmetized value of such QBFs is either 0 or 1 (rather than $O(2^{2^n})$), and allow the prover to use primes of logarithmic size.

The constant degree polynomials with logarithmic coefficients can be easily handled by weak verifiers. However, to evaluate the quantifier-free subexpressions by themselves, such verifiers need access to the random values assigned to the $O(n)$ variables, which require too much storage space. We overcome this difficulty by giving the verifiers a two-way read-only access to their random tapes, where these values are stored as consecutive blocks of $O(\log n)$ bits. □

ACKNOWLEDGMENTS. I am greatly indebted to Donald Beaver, Joan Feigenbaum, Dick Lipton, Noam Nisan, Carsten Lund, Lance Fortnow, Howard Karloff, and Laci Babai for making this result possible, and would like to thank Uri Feige and Oded Goldreich for simplifying some of my earlier proofs.

REFERENCES

1. BABAI, L. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing* (Providence, R.I., May 6–8). ACM, New York, 1985, pp. 421–429.
2. BEAVER, D., AND FEIGENBAUM, J. Hiding instances in multioracle queries. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*. Springer-Verlag, New York, 1990, pp. 37–48.
3. BOPPANA, R., HASTAD, J., AND ZACHOS, S. Does co-NP have short interactive proofs? *Inf. Proc. Lett.* 25 (1987), 127–132.
4. FORTNOW, L., AND SIPSER, M. Interactive proof systems with a log space verifier. Unpublished manuscript, 1988.

5. GAREY, M., AND JOHNSON, D. Computers and intractability, A guide to the theory of NP-completeness. W. H. Freeman, San Francisco, Calif. 1979.
6. GOLDREICH, O., MICALI, S., AND WIGDERSON, A. Proofs that yield nothing but the validity of the assertion and a methodology of cryptographic protocol design. In *Proceedings of 27th Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 174–187.
7. GOLDWASSER, S., AND SIPSER, M. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (Berkeley, Calif., May 28–30). ACM, New York, 1986, pp. 59–68.
8. GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof-systems. In *Proceedings of 17th Annual ACM Symposium on Theory of Computing* (Providence, R.I., May 6–8). ACM, New York, 1985, pp. 291–304.
9. LIPTON, R. New directions in testing. Unpublished manuscript, 1989.
10. LUND, C., FORTNOW, L., KARLOFF, H., AND NISAN, N. Algebraic methods for interactive proof systems. In *Proceedings of 31st Symposium on Foundations of Computer Science*. IEEE, New York, 1990, pp. 2–90.
11. TODA, S. On the computational power of PP and $\oplus$P. In *Proceedings of 30th Symposium on Foundations of Computer Science*. IEEE, New York, 1989, pp. 514–519.
12. VALIANT, L. The complexity of computing the permanent. *Theoret. Comput. Sci. 8* (1979), 189–201.

RECEIVED NOVEMBER 1990; REVISED AUGUST 1991; ACCEPTED SEPTEMBER 1991