Algebraic Methods for Interactive Proof Systems

Carsten Lund*
Lance Fortnow†
Howard Karloff‡
University of Chicago

Noam Nisan§
Hebrew University

Abstract

We present a new algebraic technique for the construction of interactive proof systems. We use our technique to prove that every language in the polynomial-time hierarchy has an interactive proof system. This technique played a pivotal role in the recent proofs that IP=PSPACE [S] and that MIP=NEXP [BFL].

1 Introduction

NP can be viewed as the set of languages L with this property: there is a deterministic polynomial-time verifier (Vanna) and an infinitely-powerful prover (Pat) such that for all x, if x is in L, then in polynomial time Pat can persuade Vanna that x is in L, and if x is not in L, then no prover (Pat or any other) can persuade Vanna that x *is* in L. Pat and Vanna communicate on a two-way channel (though two-way communication is not necessary here). For example, Pat can convince Vanna that a graph G is 3-colorable by exhibiting a 3-coloring. If G is not 3-colorable, no prover will ever succeed in persuading Vanna that G is 3-colorable. (Of course, co-NP-complete languages are not thought to be in NP. No prover is known who can convince a skeptical deterministic verifier that G is not 3-colorable, if it is not 3-colorable.)

We can extend this idea of "provability" by allowing Vanna to flip coins and by requiring instead that if x is in L, with probability at least 2/3 Pat persuades Vanna that x is in L, and if x is not in L, no prover can convince Vanna that x is in L with probability more than 1/3. Babai [B] and Goldwasser, Micali and Rackoff [GMR] developed this *interactive proof system* model. A summary of previous results on interactive proof systems can be found in [BM].

While certain problems such as graph non-isomorphism which are not known to be in NP were known to have interactive proof systems [GMW], theoretical computer scientists generally believed that the class

*Supported by a fellowship from the University of Århus.
†Supported by NSF grant CCR-9009936.
‡Supported by NSF grant CCR-8807534.
§Some of this work was performed at MIT and supported by NSF grant CCR-865727 and ARO grant DLL03-86-K-017.

IP of languages accepted by interactive proof systems was not much larger than NP. In particular, it was believed that co-NP-complete languages did not have interactive proof systems.

We prove that interactive proof systems have far greater power than originally believed. Our main result is an interactive proof system for the language $\{(A, s)|s$ is the permanent of 0-1 matrix $A\}$. When combined with the fact that the permanent of 0-1 matrices is #P-complete [V] and the fact that #P is hard for the polynomial-time hierarchy [T], the existence of an interactive proof system for the permanent implies that every language in the polynomial-time hierarchy has an interactive proof system. In particular, this means that every language in co-NP has an interactive proof system, even the complement of 3-COLORABILITY, for example.

For the proof we develop a new technique for reducing the problem of verifying the value of a low-degree polynomial at two given points to verifying the value at one new point. Shamir [S] has used this technique to prove that all languages in PSPACE have interactive proof systems. From the fact that IP⊆PSPACE [F], it follows that IP=PSPACE. Babai, Fortnow and Lund [BFL] have also used this technique in their proof that every language in nondeterministic exponential time has a two-prover interactive proof system in which the provers cannot communicate with one another.

Our results also have implications for program checking, verification and self-correction in the context of Blum and Kannan [BK], Blum, Luby and Rubinfeld [BLR] and Lipton [L]. In fact, the Blum-Luby-Rubinfeld and Lipton papers inspired our result.

Our result does not relativize. Fortnow and Sipser [FS] have created an oracle under which co-NP does not have an interactive proof system. To our knowledge this is the first result to "go contrary" to a previously-published oracle. Subsequent to the announcement of our result, Chor, Goldreich and Håstad [CGH] proved the same relativized result for a random oracle.

2 Definitions

A *verifier* V is a polynomial-time, probabilistic Turing machine with a special communication tape. A *prover* P is an arbitrary map f from each finite sequence $x, q_1, q_2, q_3, \ldots$, where $x \in \{0,1\}^*$ and each $q_i \in \{0,1\}^*$, to a 0-1 string.

The computation proceeds as follows. Both P and V get $x \in \{0,1\}^*$. V then computes for a while, and writes a query $q_1 \in \{0,1\}^*$ on her communication tape. P responds by replacing the q_1 with $f(x, q_1)$. V computes, overwrites $f(x, q_1)$ with a query $q_2 \in \{0,1\}^*$, and awaits P's response, $f(x, q_1, q_2)$. This process continues until V halts and accepts or rejects x. A *round* is a query from V followed by a response from P.

The pair (P, V) forms an *interactive proof system* for a language L if for all $x \in \{0,1\}^*$:

1. If $x \in L$, then $\Pr(V$ accepts input x when interacting with $P) \geq \frac{2}{3}$.
2. If $x \notin L$, then for all provers P', $\Pr(V$ accepts x when interacting with $P') \leq \frac{1}{3}$.

IP is the class of all languages which have interactive proof systems.

The class #P consists of all functions $f : \{0,1\}^* \to \mathbb{N}$ for which there exists a polynomial-time, nondeterministic Turing machine M such that for all inputs x, the number of accepting computations of M on x equals $f(x)$. $\mathrm{P}^{\#\mathrm{P}}$ is the class of languages recognized by a polynomial-time oracle Turing machine with an oracle for some function f in #P. Given x, the oracle Turing machine can learn $f(x)$ in one time step by querying its oracle.

3 The Protocol

We will prove

Theorem 1. Every language in $\mathrm{P}^{\#\mathrm{P}}$ has an interactive proof system.

Together with Toda's result that $\mathrm{P}^{\#\mathrm{P}}$ contains all the languages of the polynomial-time hierarchy [T], Theorem 1 implies

Corollary 2. Every language in the polynomial-time hierarchy has an interactive proof system. In particular, every language in co-NP has an interactive proof system.

We list some facts about the permanent of a matrix A that will be crucial in the proof of Theorem 1. If $A = (a_{ij})$ is $r \times r$, the *permanent* $\mathrm{per}(A) = \sum_{\sigma} a_{1\sigma(1)} a_{2\sigma(2)} \cdots a_{r\sigma(r)}$, where the sum is over all permutations σ of $\{1, 2, ..., r\}$. We can equivalently define the permanent recursively as $\mathrm{per}(A) = \sum_{1 \le i \le r} a_{1i} \cdot \mathrm{per}(A_{1|i})$ where $A_{1|i}$, the $(1, i)$-minor of A, is the matrix A without the first row and the ith column. The number of perfect matchings in an N-boy, N-girl bipartite graph G is equal to the permanent of G's adjacency matrix.

We will exhibit an interactive proof system for verifying the permanent of a 0-1 matrix. The following lemma implies that this is sufficient to prove Theorem 1.

Lemma 3. If $L = \{(A, s) | A$ is a 0-1 matrix and $\mathrm{per}(A) = s\}$ has an interactive proof system, then every language in $\mathrm{P}^{\#\mathrm{P}}$ has an interactive proof system.

Proof Sketch. From the fact that computing the permanent of 0-1 matrices is #P-complete [V], we can reduce the membership problem for a language $L' \in \mathrm{P}^{\#\mathrm{P}}$ to that of verifying the permanents of 0-1 matrices. Given an interactive proof system for L, it is easy to construct one for L'. ■

Throughout most of this paper we will work with the permanent over $\mathbb{Z}_p$ of an $N \times N$ matrix A with entries in $\mathbb{Z}_p$, where p is a prime in $(N!, 2N!)$. (Bertrand's Postulate [NZ] guarantees the existence of such a prime.) If A is 0-1, then the permanent of A over $\mathbb{Z}_p$ coincides with its permanent as an integer matrix, since the permanent of an $N \times N$ 0-1 matrix cannot exceed $N!$. We use the crucial fact that if B is an $r \times r$ matrix over $\mathbb{Z}_p$ whose entries are linear polynomials over $\mathbb{Z}_p$, then $\mathrm{per}(B)$ is a polynomial of degree at most r over $\mathbb{Z}_p$. Compared to p, any $r \le N$ is minuscule.

The verifier Vanna will use this fact to "trip up" a cheating prover. She will maintain a list of pairs $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_t, q_t)>$, where the B_i's are square matrices of the same size and $q_i \in \mathbb{Z}_p$. Initially $\mathcal{L} = <(A, s)>$. If $s = \mathrm{per}(A)$, then a prover who truthfully answers all of Vanna's questions will induce Vanna eventually to shrink the list to a single pair (B, q), where B is 1×1 and $q = \mathrm{per}(B)$. At that point Vanna will correctly accept the input.

If $s \ne \mathrm{per}(A)$, then however the prover answers Vanna's questions, with very high probability Vanna will maintain this "invariant": the list contains at least one pair (B_i, q_i) such that $q_i \ne \mathrm{per}(B_i)$. ("Invariant" appears in quotes because with extremely low probability, at some point every q_i might equal $\mathrm{per}(B_i)$.) When the list shrinks to one pair (B, q) where B is 1×1 and $q \ne \mathrm{per}(B)$, Vanna will reject the input (if not earlier).

How Vanna manipulates the list is the crux of the protocol. When $\mathcal{L} = <(B, q)>$ $(B = (b_{ij}),\ 1 \leq i, j \leq r$, and $r > 1)$, for each $i = 1, 2, ..., r$, Vanna constructs the minor $B_i = B_{1|i}$, asks Pat for the permanent of B_i, and gets q_i in return. Vanna checks that $q = \sum_{i=1}^{r} b_{1i} q_i$; if not, she halts and rejects. If $q = \sum_{i=1}^{r} b_{1i} q_i$, she expands $\mathcal{L}$ by replacing $\mathcal{L}$ by $<(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$. Provided that $q \neq \mathrm{per}(B)$, $q_i \neq \mathrm{per}(B_i)$ for some i.

When the list has more than one pair, Vanna shrinks the list by replacing the first two pairs $(C, c), (D, d)$ by a new pair (E, e), in the following way. The function $f(x) = \mathrm{per}(C + x(D - C))$ is a polynomial of degree at most r over $\mathbb{Z}_p$. Vanna asks Pat for the $r + 1$ coefficients of f and constructs a polynomial g from the responses. (Or Vanna could just ask for the value of f at $r + 1$ arbitrary points and interpolate herself.) If $g(0) \neq c$ or $g(1) \neq d$, Vanna rejects.

Vanna now uniformly chooses a random $a \in \mathbb{Z}_p$,[1] sends it to Pat, constructs $E = C + a(D - C)$ and $e = g(a)$, and replaces the pairs $(C, c), (D, d)$ in $\mathcal{L}$ by the one pair (E, e). The crucial fact is that if $c \neq \mathrm{per}(C)$ or $d \neq \mathrm{per}(D)$, then with probability at least $1 - r/p$, $\mathrm{per}(E) \neq e$. This follows from Lemma 4.

Lemma 4. Let C and D be $r \times r$ matrices over $\mathbb{Z}_p$. Let g be a polynomial of degree at most r over $\mathbb{Z}_p$ such that either $g(0) \neq \mathrm{per}(C)$ or $g(1) \neq \mathrm{per}(D)$. Then if a is chosen uniformly at random from $\mathbb{Z}_p$,

$$\Pr[\mathrm{per}(C + a(D - C)) = g(a)] \leq \frac{r}{p}.$$

Proof. Let $f(x) = \mathrm{per}(C + x(D - C))$, a polynomial of degree at most r over $\mathbb{Z}_p$. Since $f(0) = \mathrm{per}(C)$ and $f(1) = \mathrm{per}(D)$, clearly $f \neq g$. But two nonidentical polynomials of degree at most r over $\mathbb{Z}_p$ can coincide on at most r points. It follows that there are at most r values a such that

$$g(a) = f(a) = \mathrm{per}(C + a(D - C)). \blacksquare$$

If $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_t, q_t)>$ and at least one i satisfies $\mathrm{per}(B_i) \neq q_i$, then with very high probability, after $t - 1$ shrinkings $\mathcal{L}$ will consist of one pair (H, h) with $h \neq \mathrm{per}(H)$. The idea, then, is to replace the initial list $\mathcal{L} = <(A, s)>$ by lists of smaller and smaller matrices, until eventually $\mathcal{L} = <(B, q)>$ where B is 1×1. If $q \neq \mathrm{per}(B)$—a condition Vanna can easily test—Vanna will reject. Otherwise she'll accept.

How likely is it that Vanna will be able to maintain the "invariant?" A sequence of one expansion step followed by $r - 1$ shrinking steps will replace $\mathcal{L} = <(B, q)>$, where B is $r \times r$, by $\mathcal{L} = <(B', q')>$, where B' is $(r - 1) \times (r - 1)$. Thus fewer than N^2 steps (of either kind) suffice to reduce $\mathcal{L} = <(A, s)>$ to $\mathcal{L} = <(B, q)>$, where B is 1×1. It follows that the probability that a cheating prover can induce Vanna to erroneously accept (A, s) is less than N^2 times the minuscule probability (at most N/p) that a given expand or shrink step first violates the "invariant."

Now we give the full proof of Theorem 1.

Proof of Theorem 1. By Lemma 3, it suffices to exhibit an interactive proof system for

$$L = \{(A, s) | A \text{ is a 0-1 matrix and } \mathrm{per}(A) = s\}.$$

[1]Throughout the paper, we will assume that Vanna can choose elements of $\mathbb{Z}_p$ uniformly at random, despite the fact that she can only pick *bits* uniformly at random. In reality, she will pick integers a uniformly at random from $\{0, 1, 2, ..., M - 1\}$, where M is the least power of two exceeding p, until one is less than p. If enough trials fail to find an a less than p, she will just halt and accept x. This increases the probability of erroneously accepting an $x \notin L$ only slightly.

Here is a formal description of the protocol. A is an $N \times N$ 0-1 matrix and $0 \leq s \leq N!$.

begin

Let $\mathcal{L} =<(A, s)>$. Pat picks an integer p in $(N!, 2N!)$ and provides a short proof to Vanna that p is prime [Pr]. All arithmetic in this protocol is done mod p.

Repeat until $\mathcal{L} =<(B, q)>$ for some 1×1 matrix B:

If $\mathcal{L} =<(B, q)>$ then do

Expand: Suppose that $B = (b_{ij})$ is $r \times r$. Vanna constructs $B_i = B_{1|i}$ for $1 \leq i \leq r$. She asks Pat for the permanents of B_i, $1 \leq i \leq r$, and gets q_i for the permanent of B_i. If $\sum_{i=1}^{r} b_{1i} q_i \neq q$, Vanna rejects. Otherwise, she sets $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$.

Else ($\mathcal{L}$ has two or more pairs) do

Shrink: Vanna chooses the first two pairs (C, c) and (D, d) from $\mathcal{L}$, asks Pat for the $r+1$ coefficients of $f(x) = \mathrm{per}(C + x(D - C))$ (where C and D are $r \times r$), and constructs $g(x)$ from them. If $g(0) \neq c$ or $g(1) \neq d$, Vanna rejects. Otherwise, she chooses a random $a \in \mathbb{Z}_p$, sends it to Pat, and replaces the pairs $(C, c), (D, d)$ in $\mathcal{L}$ by $(C + a(D - C), g(a))$.

When $\mathcal{L} =<(B, q)>$ and B is 1×1, Vanna accepts if $q = \mathrm{per}(B)$ and rejects if $q \neq \mathrm{per}(B)$.

end.

The protocol contains exactly $N - 1$ *Expand* steps and $(N-1) + (N-2) + \cdots + 2 + 1$ *Shrink* steps if Vanna accepts (A, s).

We will prove:

(1) There is a prover Pat such that for all N and all $N \times N$ 0-1 matrices A, if $s = \mathrm{per}(A)$ then $\Pr[\text{Vanna accepts } (A, s)] = 1$.

(2) If $s \neq \mathrm{per}(A)$, then for all provers, $\Pr[\text{Vanna accepts } (A, s)] < N^3/p < 1/3$ (if $N \geq 6$).

It is easy to see that a prover who truthfully answers all of Vanna's questions when $\mathrm{per}(A) = s$ induces Vanna with probability one to reduce $\mathcal{L}$ to $< (B, q)>$ with B 1×1 and $q = \mathrm{per}(B)$. At this point Vanna accepts. This completes the proof of (1).

For (2), let $s \neq \mathrm{per}(A)$. Fix any prover Pat. If Vanna accepts (A, s), then at some time $\mathcal{L} =<(B, q)>$ with $q = \mathrm{per}(B)$; initially $\mathcal{L} =<(A, s)>$ with $s \neq \mathrm{per}(A)$. We say that Pat *succeeds in iteration* m if the repeat loop is executed in full at least m times, and

$$q = \mathrm{per}(B) \text{ for all } (B, q) \text{ in } \mathcal{L}$$

first occurs just after the repeat loop has been executed exactly m times. Pat succeeds in some iteration if Vanna accepts. Since there are only $N + (N-1) + \cdots + 2 < N^2$ iterations, it suffices to prove that $\Pr[\text{Pat succeeds in iteration } m] < N/p$.

Fix an m. Without loss of generality, we may assume that Pat never induces Vanna to reject until $\mathcal{L}$ consists of only one pair, in which the matrix is 1×1. (Otherwise, we may replace Pat by another prover Pat$'$ who, instead of inducing Vanna to reject early, answers the remaining questions in a way consistent with his earlier responses, as long as possible. Against such a prover Vanna will not halt until the last stage of the protocol. The probability that Pat succeeds in iteration m is no greater than the probability that Pat$'$

does.)

Pat simply cannot succeed in an *Expand* step: if $\mathcal{L} = <(B, q)>$ with $q \neq \text{per}(B)$ becomes $\mathcal{L} = <(B_1, q_1), ..., (B_r, q_r)>$ with $q_i = \text{per}(B_i)$ for all i, Vanna immediately rejects.

If iteration m is a *Shrink* step, $\mathcal{L}$ contains (C, c) and (D, d) before the step and (E, e) afterward. If $c = \text{per}(C)$ and $d = \text{per}(D)$, then $\mathcal{L}$ contained a pair (H, h) with $h \neq \text{per}(H)$. It still does. So we may assume that either $c \neq \text{per}(C)$ or $d \neq \text{per}(D)$. In this case Lemma 4 tells us that $\Pr[e = \text{per}(E)] < N/p$. Thus

$$\Pr[\text{Pat succeeds in iteration } m] < \frac{N}{p}. \blacksquare$$

4 Extensions

The protocol above requires $\Omega(N^2)$ rounds of prover-verifier communication when the input matrix is $N \times N$. Babai has suggested the following scheme to reduce the number of rounds to $O(N)$. His idea makes it possible to shrink a list $\mathcal{L}$ with r pairs into a list $\mathcal{L}'$ with one pair in one round. For $1 \leq i \leq r$, let

$$f_i(x) = \prod_{j:j \neq i, 1 \leq j \leq r} \frac{(x-j)}{(i-j)}.$$

Note that $f_i(x)$ is a polynomial over $\mathbb{Z}_p$ of degree $r-1$ with $f_i(i) = 1$, and if $j \neq i$, then $f_i(j) = 0$. Let $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$ and define $C(x) = \sum_{i=1}^r f_i(x) B_i$. The matrix $C(x)$ has entries consisting of polynomials of degree at most $r-1$. Now $f(x) = \text{per}(C(x))$ is a polynomial of degree at most $r(r-1)$ with $f(i) = \text{per}(C(i)) = \text{per}(B_i)$ for all i, $1 \leq i \leq r$. This gives us an alternative *Shrink* procedure:

Shrink: Vanna asks Pat for the $r(r-1)+1$ coefficients of $f(x) = \text{per}(C(x))$ and constructs $g(x)$ from them. If $g(i) \neq q_i$ for some $1 \leq i \leq r$, then Vanna rejects. Otherwise, she chooses a random $a \in \mathbb{Z}_p$, sends it to Pat and replaces $\mathcal{L}$ by $<(C(a), g(a))>$.

The proof of correctness is similar to that of Theorem 1 and is omitted here.

Because the number of rounds of an interactive proof system can be reduced by a constant factor [BM], for any $\epsilon > 0$ there is a variant of our permanent protocol running in at most ϵN rounds. A bounded-round protocol for the permanent would imply that the polynomial-time hierarchy collapses, since Boppana, Håstad and Zachos [BHZ] have shown that if all co-NP languages have bounded-round protocols, then the hierarchy collapses. To our knowledge this is the first example of an interactive proof system that appears to require an unbounded number of rounds.

Babai and Fortnow [BF] and Shamir [S] have provided alternate interactive proof systems for verifying the values of #P functions by counting the number of assignments satisfying a CNF formula, thus circumventing the need for Valiant's result on the completeness of the permanent. They have shown how to "arithmetize" a formula as a low-degree polynomial so that Pat and Vanna can use a protocol similar to that of Section 3 to verify the number of satisfying assignments.

Shamir [S] has shown how to arithmetize a QBF formula, using dummy variables to keep the degree low. Using a protocol similar to that in Section 3, he then proves that every language in PSPACE has an interactive proof system. Shen [Sh] later provided a "degree-reduction operator" as an alternate technique to keep the degree low.

Babai, Fortnow and Lund [BFL] have applied the techniques of this paper to multiple-prover interactive proof systems, defined by Ben-Or, Goldwasser, Kilian and Wigderson [BGKW] as interactive proof systems having a polynomial number of provers unable to communicate among themselves or to see the conversation between any other prover and the verifier. Babai, Fortnow and Lund have proven that any language computable in nondeterministic exponential time has a multiple-prover interactive proof system. Their proof uses ideas similar to those of [BF] and [S] in order to reduce the problem to that of testing the multilinearity of a function.

Cai, Condon and Lipton [CCL] have used the protocols of this paper and of Shamir [S] to prove that every PSPACE language has a bounded-round multiple-prover interactive proof system.

Fortnow and Lund [FL] have extended the techniques from this and Shamir's paper [S] to exhibit a polynomial equivalence between time-space complexity of alternating Turing machines and the time-space complexity of the verifier in a public-coin interactive proof system. In particular, they prove that every language in NC has a interactive proof system with a public-coin, polynomial-time, logarithmic-space verifier.

5 Implications

Goldwasser and Sipser [GS] have shown that one can convert any interactive proof system to one in which the verifier uses public coins, i.e., the verifier juxtaposes her coin tosses and her query message q_i on her communication tape. Furer, Goldreich, Mansour, Sipser and Zachos [FGMSZ] have shown how to modify an interactive proof system so that for true instances the verifier is convinced with probability one. Both of these properties already hold for our protocol.

Some simple corollaries that follow from these results:

Corollary 5. If cryptographic one-way functions exist then every language in the polynomial-time hierarchy has a zero-knowledge interactive proof system.

Proof. Every language with an interactive proof system has a zero-knowledge interactive proof system if one-way functions exist [BGGHKMR, IY]. ■

From Shamir [S], we infer that all languages computable in polynomial space have zero-knowledge interactive proof systems if cryptographic one-way functions exist.

Corollary 6. If every language in IP has a bounded-round interactive proof system, then the polynomial-time hierarchy collapses.

This is immediate from Boppana, Håstad and Zachos [BHZ]. Previously Aiello, Goldwasser and Håstad [AGH] constructed an oracle relative to which the class of languages with unbounded-round interactive proof systems differs from those with bounded-round interactive proof systems.

Our theorem also has applications to program checking, verification and self-correction. Lipton [L], using ideas of Beaver and Feigenbaum [BeF], showed that the permanent function can be "tested." Our protocols extend this idea and show the permanent has a self-testing/correcting pair [BLR], a pair of functions the first of which verifies that a program computes the permanent correctly on most inputs and the second of which converts a program that passes the first test into one that correctly computes the permanent on all inputs with high probability.

Theorem 1 also provides a program correctness checker [BK] for the permanent:

Corollary 7. There exists a probabilistic polynomial-time machine M that given access to a program P and a matrix A, will output with a high degree of confidence either "P outputs the correct value of the permanent of A" or "P does not correctly compute the permanent of some matrix."

Proof. In the proof of Theorem 1 the prover need only answer questions about the permanents of various matrices. We can have M simulate the verifier and use P as the prover. ■

A further discussion of the relationship between interactive proof systems and program testing can be found in [BFL].

MA is the class of languages accepted by an interactive proof system consisting of a single message from the prover to the verifier followed by probabilistic verification by the verifier. We can think of this class as the set of "publishable proofs," "proofs" that can be written down now and randomly verified years later without any help from the prover. Babai has proven that MA$\subseteq \Sigma_2^P \cap \Pi_2^P$ [B]. Corollary 8 implies that if #P has polynomial-size circuits, then $\mathrm{P}^{\#\mathrm{P}}$, and hence the polynomial-time hierarchy, lies in MA.

Corollary 8. If #P has polynomial-size circuits then $\mathrm{P}^{\#\mathrm{P}} \subseteq$ MA.

Proof. The prover gives the verifier a circuit computing the permanent. She uses this circuit as the prover in the protocol in Section 3. ■

A general discussion of Corollary 8 appears in [BFL] where it is shown that a similar result holds for PSPACE and EXP. Contrast Corollary 8 with the result of Karp and Lipton [KL] that if NP has polynomial-size circuits, then the polynomial-time hierarchy collapses to Σ_2^P.

6 Further Research

We have proven that every language reducible to a #P-complete problem has an interactive proof system, and thus so does every language in the polynomial-time hierarchy. In particular, every language in co-NP has an interactive proof system. However, even for a co-NP-complete language, in the protocol above the prover must answer #P-complete questions. Is there an interactive proof system for co-SAT where the prover need only answer questions about the satisfiability of CNF formulas? Such a proof system would give an instance checker for NP-complete languages.

We believe that one should study why this result does not relativize. One simple answer is that we have exhibited an interactive proof system for a very specific #P-complete function—the permanent—which is not #P-complete relative to any sufficiently complex oracle (since the permanent does not depend on the oracle). Babai and Fortnow [BF] have exhibited a simple characterization of #P functions by polynomials and have used this characterization to prove the main theorem of this paper without any reference to permanents. This algebraic formulation of #P does not hold in relativized worlds. Studying the algebraic structure of well-known complexity classes may lead to yet more exciting relationships among them.

References

[AGH] W. Aiello, S. Goldwasser, and J. Håstad. On the power of interaction. *Combinatorica*, 10(1):3–25, 1990.

[B] L. Babai. Trading group theory for randomness. In *Proc. of the 17th ACM Symp. on the Theory of Computing*, pages 421–429, 1985.

[BeF] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. 7th Symp. on Theoretical Aspects of Comp. Sci.* LNCS 415:37–48, 1990.

[BF] L. Babai and L. Fortnow. Arithmetization: a new method in structural complexity theory. *Computational Complexity*, 1:41–66, 1991.

[BFL] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BGGHKMR] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *Proc. Crypto 88*, pages 37–56, 1988.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proc. of the 20th ACM Symp. on the Theory of Computing*, pages 113–131, 1988.

[BHZ] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[BK] M. Blum and S. Kannan. Designing programs that check their work. In *Proc. of the 21st ACM Symp. on the Theory of Computing*, pages 86–97, 1989.

[BLR] M. Blum, M. Luby, and R. Rubinfeld. Self-testing and self-correcting programs, with applications to numerical programs. In *Proc. of the 22nd ACM Symp. on the Theory of Computing*, 1990.

[BM] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *J. of Computer and System Sciences*, 36(2):254–276, 1988.

[C] S. Cook. The complexity of theorem-proving procedures. In *Proc. of the 3rd ACM Symp. on the Theory of Computing*, pages 151–158, 1971.

[CCL] J. Cai, A. Condon, and R. J. Lipton. PSPACE is provable by two provers in one round. In *Proc. of the 6th Conference on Structure in Complexity Theory*, pages 110–115, 1991.

[CGH] B. Chor, O. Goldreich, and J. Håstad. The random oracle hypothesis is false. Manuscript, Technion, Haifa, Israel, 1990.

[F] P. Feldman. The optimum prover lives in PSPACE. Manuscript, M.I.T., 1986.

[FGMSZ] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 429–442. JAI Press, 1989.

[FL] L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity. In *Proc. 8th Symp. on Theoretical Aspects of Comp. Sci.* LNCS 480:263–274, 1991. To appear in *Theoretical Computer Science*.

[FRS] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. of the 3rd Conference on Structure in Complexity Theory*, pages 156–161, 1988.

[FS] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28:249–251, 1988.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. on Computing*, 18(1):186–208, 1989.

[GMW] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 174–187, 1986.

[GS] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, 1989.

[IY] R. Impagliazzo and M. Yung. Direct minimum-knowledge computation. In *Proc. Crypto 87*, pages 40–51. LNCS 293, 1987.

[KL] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. of the 12th ACM Symp. on the Theory of Computing*, pages 302–309, 1980.

[L] R. Lipton. New directions in testing. In J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. American Mathematical Society, 1991.

[NZ] I. Niven and H. S. Zuckerman. *An introduction to the theory of numbers.* John Wiley and Sons, New York, 4th edition, pages 224–225, 1980.

[Pr] V. Pratt. Every prime has a succinct certificate. *SIAM J. Comp*, 4:214–220, 1975.

[S] A. Shamir. IP=PSPACE. In *Proc. of the 31st IEEE Symp. on Foundations of Computer Science*, pages 11–15, 1990.

[Sh] A. Shen. IP=PSPACE: simplified proof. Manuscript, Institute of Problems of Information Transmission, Moscow, USSR, 1990.

[Si] J. Simon. *On some central problems in computational complexity.* PhD thesis, Cornell University, Computer Science, 1975. Tech Report TR 75-224.

[SS] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM J. of Computing*, 6:84–85, 1977. See also erratum 7:118, 1978.

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

[V] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.