On the Power of Multi-Prover Interactive Protocols

Lance Fortnow*
John Rompel†
Michael Sipser‡

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

1 Introduction

Interactive proof systems, as described in [23] and [3], are a model in which a probabilistic polynomial time verifier may interactively ask questions of a prover with unbounded computational power in order to decide the truth of a proposition. This is a generalization of the NP type proof system in which the verifier may only listen and not speak or toss coins.

In this paper we consider a further generalization of the proof system model, due to Ben-Or, Goldwasser, Kilian and Wigderson [6], where instead of a single prover there may be many. This apparently gives the model additional power. The intuition for this may be seen by considering the case of two criminal suspects who are under interrogation to see if they are guilty of together robbing a bank. Of course they (the provers) are trying to convince Scotland Yard (the verifier) of their innocence. Assuming that they are in fact innocent, it is clear that their ability to convince the police of this is enhanced if they are questioned in separate rooms and can corroborate each other's stories without communicating. We shall see later in this paper that this sort of corroboration is the key to the additional power of multiple provers.

Interactive proof systems have seen a number of important applications to cryptography [23, 22], algebraic complexity [3], program testing [7, 8] and distributed computation [16, 23]. For example, a chain of results concerning interactive proof systems [22, 3, 24, 9] conclude that if the graph isomorphism problem is NP-complete then the polynomial time hierarchy collapses. Multiple-prover interactive proof systems have also seen several important applications including the analysis of program testing [7, 4] and the complexity of approximation algorithms [14, 2, 1].

Brief summary of results: First we give a simple characterization of the power of the multi-prover model in terms of probabilistic oracle Turing machines. Then we show that every language accepted by multiple prover interactive proof systems can be computed in nondeterministic exponential time. Babai, Fortnow and Lund [4] have since shown this bound is tight. We then show results like the one proved by Babai, Fortnow and Lund can not relativize by exhibiting an oracle relative to which there exist co-NP problems that do not have multiple prover interactive proof systems. We show, however,

*supported by an Office of Naval Research fellowship.
Current Address: Computer Science Dept., University of Chicago, 1100 E. 58th St., Chicago, IL 60637
†supported by National Science Foundation Fellowship and the third author's grants.
Current Address: D.E. Shaw & Co., 251 Park Avenue South, New York, NY 10010
‡supported by NSF Grant DCR-8602062 and Air Force Grant AFOSR-86-0078.

that the existence of an oracle relative to which there exist languages with multiple prover interactive proof systems but can not be computed in polynomial space would imply an unrelativized separation of NP and poly-log space. Finally, we show a simple example that illustrates that multiple prover interactive proof systems do not behave independently in parallel as previously believed.

2 Definitions and Other Results

Let $P_1, P_2, \ldots P_k$ be infinitely powerful machines and V be a probabilistic polynomial-time machine, all of which share the same read-only input tape. The verifier V shares communication tapes with each P_i, but different provers P_i and P_j have no tapes they can both access besides the input tape. We allow k to be as large as a polynomial in the size of the input; any larger and V could not access all the provers.

Formally, each P_i is a function from the input and the conversation it has seen so far to a message. We put no restrictions on the complexity of this function other than that the lengths of the messages produced by this function must be bounded by a polynomial in the size of the input. We will assume throughout this paper that the inputs are drawn from the set of strings over the alphabet $\Sigma = \{0, 1\}$.

$P_1, \ldots, P_k$ and V form a multi-prover interactive protocol for a language L if:

1. If $x \in L$ then $\Pr(P_1, \ldots, P_k \text{ and } V \text{ on } x \text{ accept}) > 1 - 2^{-n}$.
2. If $x \notin L$ then for all provers $P'_1, \ldots, P'_k$, $\Pr(P'_1, \ldots, P'_k \text{ and } V \text{ on } x \text{ accept}) < 2^{-n}$

MIP is the class of all languages which have multi-prover interactive protocols. If k is one we get the class IP of languages accepted by one-prover interactive proof systems [23].

Note that we require an exponentially small probability of error. We could reduce a constant error to a probability of error of less than $2^{-p(n)}$ for any polynomial $p(n)$ by running the protocols several times serially. Unlike the result of Babai and Moran [5] for the one-prover model, it is unknown whether we can decrease the probability of error in multi-prover proof systems by running the protocols in parallel (see section 6).

A *round* of an multi-prover interactive protocol consists of messages from the verifier to some or all of the provers followed by messages from these provers to the verifier. In general, interactive protocols can have a polynomial number of rounds. We let α_{ij} designate a message from prover i to the verifier in round j and β_{ij} designate a message from the verifier to prover i in round j.

Ben-Or, Goldwasser, Kilian and Wigderson [6] originally developed multi-prover interactive proof systems primarily for cryptographic purposes. They show every language accepted by a two prover interactive proof system has a perfect zero-knowledge two prover proof system, where even NP does not have perfect zero-knowledge single prover proof systems unless the polynomial-time hierarchy collapses [17]. They also show two prover systems can simulate any multi-prover system. Along the lines of Furer, Goldreich, Mansour, Sipser and Zachos [21], they show any two prover system has an equivalent system that accepts with probability one for strings in the language. Complete proofs of these results appeared in [25].

Subsequent to the results described in this paper, the complexity of interactive proof systems have been shown to be much more powerful then previously believed. Lund, Fortnow, Karloff and Nisan [27] have shown the existence of an interactive proof system for every language in the polynomial time hierarchy. Using the techniques of Lund, Fortnow, Karloff and Nisan, Shamir [29] has shown that every language computable in polynomial space has an interactive proof system. Building on the result of Lund, Fortnow, Karloff and Nisan and Theorem 3.1, Babai, Fortnow and Lund [4] have shown that every language accepted in nondeterministic exponential time has a two-prover interactive proof system.

A series of results due to Cai, Condon, Lipton, Lapidot, Shamir, Feige and Lovász [12, 10, 11, 13, 26, 15] have modified the protocol of Babai, Fortnow and Lund [4] to show that every language in NEXP has a two-prover, one-round proof systems with an exponentially small error, strengthening the (unproven) claims made in an earlier version of this paper [19]. The general question of parallelizing protocols remains open (see [15]).

Feige, Goldwasser, Lovász, Safra and Szegedy [14] use the Babai-Fortnow-Lund [4] result to prove some grave consequences of clique approximation. Arora and Safra [2] improved these results and Arora, Lund, Motwani, Sudan and Szegedy [1] applied these techniques to the MAX SNP-hard problems [28].

3 Probabilistic Oracle Machines

Suppose a prover in an interactive proof system must set all his possible responses before the protocol with the verifier takes place. We can think of the prover as an oracle attempting to convince a probabilistic machine whether to accept a certain input string. The oracle must be fully specified before the protocol begins.

Let M be a probabilistic polynomial-time Turing machine with access to an oracle O. A language L is accepted by an oracle machine M iff

1. For every $x \in L$ there is an oracle O such that M^O accepts x with probability $> 1 - 2^{-n}$
2. For every $x \notin L$ and for all oracles O', $\mathrm{M}^{O'}$ accepts with probability $< 2^{-n}$

This model differs from the one-prover interactive protocol model in that the oracle must be set ahead of time while in an interactive protocol the prover may let his future answers depend on previous ones.

Theorem 3.1 *L is accepted by an probabilistic oracle machine if and only if L is accepted by a multi-prover interactive protocol.*

Proof

($\Longleftarrow$)

Suppose L is accepted by a multi-prover interactive proof system V. Without loss of generality, we can assume that all messages from the provers to the verifier consist of only a single bit. Then define M as follows: M simulates V with M remembering all messages. When V sends a message to a prover, M asks the oracle the question $(x, i, j, \beta_{i1}, \ldots, \beta_{ij})$ suitably encoded and uses the response as the j^{th} message from prover i on input x where $\beta_{i1}, \ldots, \beta_{ij}$ are the first j messages sent from the verifier to prover i. M then accepts x if and only if V does.

1. Let $P_1, \ldots, P_k$ be provers which cause V to accept each $x \in L$ with probability at least $1 - 2^{-n}$. If we let O be the oracle which encodes in the above manner the messages of $P_1, \ldots, P_k$, then M^O will accept each $x \in L$ with the same probability as V.
2. Suppose there were an input $x \notin L$ and an oracle O' such that $M^{O'}$ accepts x with probability more than 2^{-n}. Then we could construct provers $P'_1, \ldots, P'_k$ which cause V to accept x with the same probability by just using O' to create their messages. Since, by definition, no such $P'_1, \ldots, P'_k$ exist, neither does O'.

($\Longrightarrow$)

Suppose L is accepted by a probabilistic oracle machine M in n^k steps. We will define a verifier, V, to simulate M using $4n^{k+1}$ provers. First the verifier flips two sets of coins r_1 and r_2. The verifier

uses r_1 to choose a random ordering of the $4n^{k+1}$ provers. The verifier then simulates M using r_2 and whenever M asks an oracle question, V asks the question to each of the next n provers in the chosen ordering. If the provers are unanimous in their answer, V uses that answer in its simulation of M; if not, V rejects immediately. If the provers successfully answer all oracle queries, then the verifier accepts if and only if M does. There can be at most n^k questions so V will use at most n^{k+1} provers. There will be at least $3n^{k+1}$ unused provers.

1. Let O be an oracle such that M^O accepts each $x \in L$ with probability at least $1 - 2^{-n}$. If we let $P_1, \ldots, P_{4n^{k+1}}$ all answer (identically) according to O, then they will cause V to accept each $x \in L$ with the same probability as M^O.

2. Consider $x \notin L$; consider any provers $P'_1, \ldots, P'_{4n^{k+1}}$. Let oracle O' answer queries as the majority of $P'_1, \ldots, P'_{4n^{k+1}}$ would. If there is no majority then O' answers arbitrarily. We know that O', like every oracle, cannot cause $M^{O'}$ to accept x with probability greater than 2^{-n}. We need to also show that the provers $P'_1, \ldots, P'_{n^{k+1}}$ cannot do much better in the above defined protocol.

   There are two cases to consider for V accepting x: either all oracle queries in the simulation answered consistently with O' or some oracle query answered differently than O' would. By the definition of acceptance for probabilistic oracle machines, we know that the probability of accepting given that the first case occurs is bounded by 2^{-n}, where this probability is over the random coins of M.

   Now consider the second case. For V to accept using an oracle answer inconsistent with O', it must be the case that, for some i, the i^{th} set of n provers all give an answer inconsistent with O' on the i^{th} query. Let S_i be the event that i is least with this property given that V accepts. Fix r_2 and all the first $i-1$ sets of provers. There are $4n^{k+1} - n(i-1) > 3n^{k+1}$ remaining provers. Of these provers at most $2n^{k+1}$ can give an answers inconsistent with O' on the i^{th} query. Thus the probability that S_i occurs is bounded by

$$\frac{\binom{2n^{k+1}}{n}}{\binom{3n^{k+1}}{n}} \leq (2/3)^n .$$

   The probability that S_i will occur for some i is at most n^k times this.

   We will use the following identity for any two events B and C:

$$\Pr(B \wedge C) = \Pr(B|C)\Pr(C) \leq \Pr(B|C) \tag{1}$$

   Let A be the event that V accepts. Let F be the event that the first case occurs. Let S be the event that the second case occurs. By (1), we have:

$$\Pr(A) = \Pr(A \wedge F) + \Pr(A \wedge S) \leq \Pr(A|F) + \Pr(S|A) \leq 2^{-n} + n^k(2/3)^n < (n^k + 1)(2/3)^n.$$

We can reduce the error in the usual way by running this protocol in series several times. □

Theorem 3.1 gives a natural model equivalent to multiple provers and useful for proving theorems about them. In fact the proof that multiple provers can simulate nondeterministic exponential time [4] requires this theorem. We can also make connections to program checking.

Blum and Kannan [7] define *function-restricted IP* as the set of languages accepted by a probabilistic oracle machine with the additional restriction on the first requirement:

1. For every $x \in L$, M^L accepts x with probability $> 1 - 2^{-n}$

In other words, the "honest oracle" must just compute the language but the "dishonest" oracle may still compute any function.

Blum and Kannan also define an *instance checker* $C_L^{\mathcal{P}}$ for a language L and an instance $x \in \{0,1\}^*$ as a probabilistic polynomial-time oracle Turing Machine that given a program $\mathcal{P}$ claiming to compute L, and an input x:

1. If $\mathcal{P}$ correctly computes L for all inputs then with high probability $C_L^{\mathcal{P}}$ will output "correct".
2. If $\mathcal{P}(x) \neq L(x)$, with high probability $C_L^{\mathcal{P}}(x)$ will output "$\mathcal{P}$ does not compute L".

Blum and Kannan show that a language L has an instance checker if both L and $\overline{L}$ have function-restricted interactive proof systems.

The proof of Theorem 3.1 yields the following corollary:

Corollary 3.2 *A language L has a function restricted interactive proof system if and only if there exists a multiple prover interactive proof system for L where the honest provers need only answer questions about L.*

Arora and Safra [2] define a hierarchy of complexity classes PCP (for probabilistically checkable proofs), corresponding to the number of random and query bits required to verify a proof of membership in the language, as follows:

A *verifier* M is a probabilistic polynomial-time Turing machine with random access to a string Π representing a membership proof; M can *query* any bit of Π. Call M an $(r(n), q(n))$-*restricted* verifier if, on an input of size n, it is allowed to use at most $r(n)$ random bits for its computation, and query at most $q(n)$ bits of the proof.

A language L is in $\mathrm{PCP}(r(n), q(n))$ if there exists an $(r(n), q(n))$-restricted verifier M such that for every input x:

1. If $x \in L$, there is a proof Π_x which causes M to accept for every random string, i.e.,with probability 1.
2. If $x \notin L$, then for all proofs Π, the probability over random strings of length $r(n)$ that M using proof Π accepts is bounded by $1/2$.

Notice that the role of Π is identical to the role of the oracle O in our definition of probabilistic oracle machines. Thus combining Theorem 3.1 with the fact that any multiple-prover interactive proof system has an equivalent system that accepts with probability one for strings in the language [6, 21] we have the following corollary:

Corollary 3.3 *$MIP = \cup_{k>0} PCP(n^k, n^k)$.*

Thus Babai, Fortnow and Lund [4] show that $\mathrm{NEXP} = \cup_{k>0}\mathrm{PCP}(n^k, n^k)$. Arora, Lund, Motwani, Sudan and Szegedy [1] show that $\mathrm{NP} = \cup_{c>0}\mathrm{PCP}(c\log(n), c)$.

4 Nondeterministic Exponential Time Suffices

In this section, we show an upper bound on the complexity of multiple prover interactive proof systems.

Theorem 4.1 *If there exists a multiple prover interactive proof system accepting a language L then L can be computed in nondeterministic exponential time.*

By nondeterministic exponential time, we mean $\cup_{k>0}\text{NTIME}[2^{n^k}]$.

Proof By Theorem 3.1, we can assume there exists a probabilistic oracle machine M accepting L with M using time n^k on inputs of length n for some $k > 0$. We create a nondeterministic exponential time machine to accept L as follows: On input x of length n, guess the value of the oracle O on all questions of length at most n^k. Note $M(x)$ can only ask oracle questions of length no longer than n^k. There are exactly $2^{n^k+1} - 1$ such questions. For r a string of length n^k, let $f(x, O, r) = 1$ if M on input x accepts using random coin tosses r and getting the oracle answers from O and $f(x, O, r) = 0$ otherwise. Compute

$$S = \sum_{r \in \{0,1\}^{n^k}} f(x, O, r).$$

Accept if $S > 2^{n^k}/2$.

By the definition of probabilistic oracle machines, for $x \in L$ there exists a setting of the oracle such that $S \geq (1 - 2^{-n})2^{n^k}$. If $x \notin L$ then for any setting of the oracle, $S \leq 2^{-n}2^{n^k}$. This proves the correctness of the computation above. □

Babai, Fortnow and Lund [4] have shown that any language computable in nondeterministic exponential time has a multiple prover interactive proof system. Thus we have an equivalence of the class of languages provable by multiple prover proof systems and those computable in nondeterministic exponential time.

Note Theorem 4.1 does not show that any language L with a multiple prover proof system can be proven with provers of nondeterministic exponential time complexity. The provers must actually find the nondeterministic guesses that would make the nondeterministic exponential time machine accept. This would require the second level of the exponential time hierarchy to determine, say, the lexicographically first such series of nondeterministic guesses. We do not know whether they can be improved to have nondeterministic exponential time complexity. Babai, Fortnow and Lund [4] show any language in deterministic exponential time requires only deterministic exponential time provers.

5 Relativized Limits on Multiple Provers

The result of Babai, Fortnow and Lund [4] that shows all languages computable in nondeterministic exponential time have multiple prover interactive proof systems does not relativize, i.e.,their proof does not imply that given any oracle A, nondeterministic exponential time with access to oracle A has a multiple prover interactive proof with the provers and verifiers also having access to oracle A. We show that any proof of this result can not relativize:

Theorem 5.1 *There exists an oracle A and a language $L \in$ co-NP^A such that $L \notin$ MIP^A.*

Theorem 5.1 extends a result by Fortnow and Sipser [20] that shows the existence of an oracle relative to which co-NP does not have single prover interactive proofs.

Proof In this proof we will use the oracle machine model. It is easy to verify that the proof of Theorem 3.1 holds under relativizations to all oracles. Note that our machines can ask questions about two oracles, the "prover" oracle O and the "relativization" oracle A.

We can enumerate all possible probabilistic polynomial-time machines in the standard manner, letting M_i be bounded in time by n^i, where n is the size of the input.

For any oracle A, let

$$L(A) = \{1^n : A \text{ contains all strings of length } n\}.$$

It is clear that $L(A) \in \text{co-NP}^A$ for all oracles A.

In step i we make $L(A)$ different than any language accepted by oracle machine M_i^A using any prover oracle O. Then $L(A)$ can not have a multi-prover interactive protocol and we have proved our theorem.

This idea is as follows: If $1^n \in L(M^A)$ then $M^A(1^n)$ must accept with probability at least $1 - 2^{-n}$. If $1^n \notin L(M^A)$ then $M^A(1^n)$ accepts with probability at most 2^{-n}. We will pick a length n and a string x of length n such that whether $x \in A$ will determine whether $1^n \in L(A)$ but whether $x \in A$ will only affect the probability of whether $M^A(1^n)$ accepts by less than $1/2$. This will allow us to diagonalize against M^A.

STEP i: Pick N_i large enough so $2^{N_i} > 2(N_i)^i$ and no oracle questions to A of length N_i have been asked in any previous step. Let $p_i = (N_i)^i$.

Define a machine M_i' that simulates M_i^A using a built-in table of the finite locations where A has already been defined. When M_i^A queries a string from A, M_i' will either answer correctly if that string has been previously set otherwise M_i' will answer yes to that oracle question.

If there are not any oracles O such that O and M_i' accept on input 1^{N_i} with probability at least $1 - 2^{-n}$ then we put in the oracle A all strings of length N_i and every other previously unset string that M_i^A asks about for any oracle O. This completes step i. Note that M_i^A can only ask questions of length less than p_i so we will always be able to find N_{i+1} in step $i + 1$.

Otherwise we have some oracle O such that O and M_i' will accept 1^{N_i} with probability at least $1 - 2^{-n}$. On any computation path (which is determined by M_i^A's coin tosses), M_i^A can ask at most p_i oracle questions to A of length N_i. There are 2^{N_i} questions of length N_i. A counting argument shows that there is some oracle question x of length N_i that appears in no more than $p_i/2^{N_i}$ ratio of the computation paths of M_i^A. By the way we chose N_i this means the oracle question x appears in less than one half of the computation paths of M_i^A. Put all strings of length N_i except for x in the oracle A. Also place in the oracle A every string queried of any other length by M_i^A on every possible communication with every possible O. The oracle O will convince M_i^A to accept with probability greater than 2^{-n} since more than a half of the computation paths act the same as the corresponding paths of M_i'.

If there exists an oracle O that makes M_i^A accept 1^{N_i} with probability greater than $1 - 2^{-n}$ then $L(A)$ does not contain 1^{N_i}. Conversely, if no oracles exist that cause M_i^A to accept with probability at least 2^{-n} then $L(A)$ does contain 1^{N_i}. By the standard diagonalization argument $L(A)$ does not equal the language accepted by M_j^A for any j. □

This result implies the earlier result of Fortnow and Sipser [20] since the language $L(A)$ does not have one-prover interactive proof systems under the oracle A.

The result of Babai, Fortnow and Lund [4] gives us strong evidence that there exist languages not computable in polynomial space that have multiple prover interactive proof systems since most theoretical computer scientists believe polynomial space is not sufficient to accept all languages computable in nondeterministic exponential time. However, an oracle to separate MIP and PSPACE would imply a major separation result:

Theorem 5.2 *If there exists an oracle A such that PSPACE^A does not contain MIP^A then there exist languages in NP that can not be computed in poly-log space.*

Proof Assume that every language in NP can be accepted in poly-log space. Let A be any oracle and L be a language in MIP^A accepted by a verifier that runs in time n^k. From the proof of Theorem 4.1 we know that L is accepted by a nondeterministic exponential-time oracle machine $M(x)$ that only looks at the oracle queries of A of length no larger than $|x|^k$. Define $M'(y)$ where

$$y = (x, b_\epsilon b_0 b_1 b_{00} b_{01} \cdots b_{\underbrace{11\ldots1}_{|x|^k}})$$

as the machine that simulates $M(x)$ using bit b_z as the answer to oracle query z. Since $|y| = \Omega(2^{|x|^k})$, M' runs in nondeterministic polynomial time in the length of y. Thus $L(M') \in \mathrm{NP}$ and thus by assumption there exists a polylog space machine $N(y)$ that accepts $L(M')$. We create a new machine $N'(x)$ using oracle A that works as follows: Simulate $N(y)$ (without creating y) and whenever N asks about bit b_z use $A(z)$. It's easy to see that $L(N') = L$ and we can simulate N' in polynomial (in $|x|$) space with access to the oracle A. □

6 Bounded Round Protocols

In an earlier version of this paper [19], we claimed two results about collapsing rounds in multi-prover proof systems: Every language provable by a single-prover interactive proof system has a two-prover protocol using only one round and every language provable by a multi-prover proof system has a three-prover protocol using only two rounds. Unfortunately, we have since discovered an error in the "proof" of these statements. Our arguments required that we can somehow decrease the error probability of certain protocols by running them in parallel. We assumed that if the provers can be prevented from communicating among themselves through the protocol then parallel runs of the protocol work independently like parallel runs of one prover interactive protocols [5]. Unfortunately, this assumption is fallacious.

As mentioned in section 2, results of Cai, Condon, Lipton, Lapidot, Shamir, Feige and Lovász [12, 10, 11, 13, 26, 15] show that we can create a two-prover one-round protocol with exponentially small error equivalent to any multi-prover protocol. However, the proofs work by looking at special properties of the Babai-Fortnow-Lund [4] protocol instead of showing how to parallelize general multi-prover protocols.

We show the parallelization assumption faulty in even a simple case with the following counterexample. This example first appeared in [18].

Suppose we have the following two prover protocol:

V: Pick two bits a and b uniformly and independently at random.

$V \rightarrow P_1$: a

$V \rightarrow P_2$: b

$P_1 \rightarrow V$: c

$P_2 \rightarrow V$: d

V: Accept if $(a \vee c) \neq (b \vee d)$.

It is easy to show the best strategy for two provers causes the verifier to accept with probability 1/2. Notice neither prover has any notion of what bit the verifier has sent to the other prover.

Now let us examine the two round version of the same protocol:

V: Pick bits a_1, a_2 and b_1, b_2 uniformly and independently at random.

$V \rightarrow P_1$: a_1, a_2

$V \rightarrow P_2$: b_1, b_2

$P_1 \rightarrow V$: c_1, c_2

$P_2 \rightarrow V$: d_1, d_2

V: Accept if $(a_1 \vee c_1) \neq (b_1 \vee d_1)$ and $(a_2 \vee c_2) \neq (b_2 \vee d_2)$.

If the parallel runs of the protocol behave independently we would expect the optimum strategy for the provers causes the verifier to accept with probability $(1/2)^2 = 1/4$. However the following strategy for the provers causes the verifier to accept with probability $3/8$:

P_1: If $a_1 = a_2 = 0$ respond $c_1 = c_2 = 0$ otherwise respond $c_1 = c_2 = 1$.

P_2: If $b_1 = b_2 = 0$ respond $d_1 = d_2 = 0$ otherwise respond $d_1 = d_2 = 1$.

Note in n rounds the probability of acceptance of this protocol can not exceed $(3/4)^n$ since the verifier will not accept if $a_i = b_i = 1$ for any i. We can not find any counterexample without this type of exponential decrease. However we have not been able to prove any such decrease in a general setting.

7 Further Research

There still remain many open questions including:

- What effect does running protocols in parallel have? In particular, if a protocol is run in parallel for m rounds, is the error necessarily exponentially small in m?
- A public-coin interactive proof system can accept any language accepted by a interactive proof system [24]. What can we say about public-coin multi-prover interactive proof systems? How do we even define public-coin proof systems for multiple provers?
- Do there exists multiple prover interactive proof systems for proving co-NP questions where the provers need only answer NP questions? A positive result would imply an instance checker for NP-complete problems.

Acknowledgment

We would like to thank Yishay Mansour for his help with Theorem 5.2. We would also like to thank the anonymous referees for their various helpful suggestions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23. IEEE, New York, 1992.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 2–13. IEEE, New York, 1992.

[3] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, New York, 1985.

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

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

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

[7] M. Blum and S. Kannan. Designing programs that check their work. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 86–97. ACM, New York, 1989.

[8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing and self-correcting programs, with applications to numerical programs. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, pages 73–83. ACM, New York, 1990.

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

[10] J. Cai, A. Condon, and R. Lipton. On bounded round multi-prover interactive proof systems. In *Proceedings of the 5th IEEE Structure in Complexity Theory Conference*, pages 45–54. IEEE, New York, 1990.

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

[12] J. Cai, A. Condon, and R. Lipton. On games of incomplete information. *Theoretical Computer Science*, 103(1):25–38, 1992.

[13] U. Feige. On the success probability of the two provers in one round proof systems. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 116–123. IEEE, New York, 1991.

[14] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 2–12. IEEE, New York, 1991.

[15] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 733–744. ACM, New York, 1992.

[16] P. Feldman and S. Micali. From scratch to byzantine agreement in constant expected time. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 148–161. ACM, New York, 1988.

[17] L. Fortnow. The complexity of perfect zero-knowledge. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 327–343. JAI Press, Greenwich, 1989.

[18] L. Fortnow. *Complexity-theoretic aspects of interactive proof systems.* PhD thesis, Massachusetts Institute of Technology, May 1989. Tech Report MIT/LCS/TR-447.

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

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

[21] 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, Greenwich, 1989.

[22] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

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

[24] 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, Greenwich, 1989.

[25] J. Kilian. *Uses of Randomness in Algorithms and Protocols*. ACM Distinguised Dissertation. MIT Press, Cambridge, Massachusetts, 1990.

[26] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-time. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 13–18. IEEE, New York, 1991.

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

[28] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[29] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.