# On Completeness and Soundness in Interactive Proof Systems

*Martin Furer*, Computer Science Dept., Pennsylvania state Univ., University Park, PA 16802.

*Oded Goldreich*, Computer Science Dept., Technion, Haifa, Israel.

*Yishay Mansour*, Lab. for Computer Science, MIT, Cambridge, MA 02139.

*Michael Sipser*, Mathematics Dept., MIT, Cambridge, MA 02139.

*Stathis Zachos*, Comp. and Inform. Sci., Brookline College of CUNY, Brookline, NY 11210.

**ABSTRACT** – An interactive proof system with *Perfect Completeness* (resp. *Perfect Soundness*) for a language $L$ is an interactive proof (for $L$) in which for every $x \in L$ (resp. $x \notin L$) the verifier *always* accepts (resp. *always* rejects). We show that any language having an interactive proof system has one (of the Arthur-Merlin type) with perfect completeness. On the other hand, only languages in $NP$ have interactive proofs with perfect soundness.

Work done while third author was working at the IBM-Scientific Center, Technion City, Haifa, Israel. Second author was partially supported by the Fund for Basic Research Administered by the Israeli Academy of Sciences and Humanities. Fifth author was partially supported by PSC-CUNY grant.

Appeared in *Advances in Computing Research: A Research Annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.

**Warning:** Reproduced almost automatically from an old troff file. The resulting text was not proofread.

**Updated affiliation** for Oded Goldreich – Department of Computer Science and Applied Mathematics Weizmann Institute of Science, Rehovot, ISRAEL. Email: `oded@wisdom.weizmann.ac.il`

## 1. INTRODUCTION

The two basic notions regarding a proof system are *completeness* and *soundness*. Completeness means that the proof system is powerful enough to generate "proofs" for all the valid statements (in some class). Soundness means that any statement that can be proved is valid (i.e. no "proofs" exist for false statements). Two computational tasks related to a proof system are generating a proof and verifying the validity of a proof. This naturally suggests the notions of a *prover* (a party able of generating proofs) and a *verifier* (a party capable of validating proofs). Typically, the verifier's task is easier than the prover's task. In order to focus on the complexity of the verification task it is convenient to assume that the prover has unlimited power. For many years $NP$ was considered *the formulation* of *"whatever can be efficiently verified"*. This stemmed from the association of deterministic polynomial-time computation with efficient computation. The growing acceptability of probabilistic polynomial-time computations as reflecting efficient computations is the basis of more recent formalizations of "whatever can be efficiently verified". In these formalizations, due to Goldwasser, Micali and Rackoff [GMR] and Babai [B], and shown to be equivalent by Goldwasser and Sipser [GS], the (polynomial time) verifier is allowed to toss coins and arbitrarily interact with the prover, furthermore he can accept or reject based on overwhelming statistical evidence. Ruling by overwhelming statistical evidence means relaxing the completeness and soundness conditions so that any valid statement can be proved with a very high probability while any false statement has only negligible probability to be proved. For a definition of *interactive proof systems* we refer the reader to Goldwasser and Sipser's article in this volume [GS]. We denote by *IP* the class of languages for which there exists an interactive proof system. Clearly, $NP \subset IP \subset PSPACE$. It is believed that the class $NP$ is *strictly* contained in $IP$. Evidence for this may perhaps be derived from the fact that, relative to some oracle, interactive proofs are even not contained in the polynomial-time hierarchy, i.e. $\exists A$ s.t. $IP^A - PH^A \neq \emptyset$ (see [AGH]). It is also interesting to note that natural languages as Graph Non-Isomorphism and Matrix Group Non-Membership, which are not known to be in $NP$, where shown to be in $IP$ (by [GMW] and [B], respectively). Considering an interactive proof system, it seems that in some sense the prover is "responsible" for the completeness condition, while the verifier is "responsible" for the soundness condition. If this intuition is correct, and the prover has unrestricted power, why should the completeness condition be relaxed? Namely, can one modify the interactive proof such that the prover never fails in demonstrating the validity of true statements, while maintaining soundness. By *perfect completeness* we mean that the prover *never* fails to prove the membership of inputs that are indeed in the language, while *perfect soundness* means that the verifier *never* accepts inputs that are not in the language. Perfect completeness and perfect soundness are not only theoretically interesting, but are also of practical importance. This is the case, since probabilistic completeness and soundness are defined with respect to ideal (unbiased) coin tosses and may not hold when using pseudorandom sequences (even in the sense of Blum and Micali [BM] and Yao [Y]). On the other hand perfect completeness and soundness are independent of the quality of the verifier coin tosses. Our main result is that *Interactive Proofs with Perfect Completeness* **are as powerful** *as Interactive Proofs*. The proof of the main result is in fact a transformation that given an interactive proof for a language $L$ yield an Arthur-Merlin interactive proof with perfect completeness for $L$. This transformation preserves the number of interactions of the original interactive proof. An alternative proof which uses different ideas, and in particular a protocol for "random selection" appears in [GMS]. An alternative characterization of complexity classes define by bounded Arthur-Merlin games was presented in [ZF]. They use polynomially bounded quantifiers $\exists, \forall, \exists^+$ (where $\exists^+$ means roughly "for most"). For all quantifier strings $Q_1$, $Q_2$ of equal length over $\{\exists, \forall, \exists^+\}$ the notation $(Q_1/Q_2)$ represents the classes of languages satisfying:

- $x \in L \rightarrow Q_1\bar{y}\ \ P(x, \bar{y})$
- $x \notin L \rightarrow Q_2\bar{y} \neg P(x, \bar{y})$

for some poly-time computable predicate $P$. In this notation $(\exists^+\exists/\exists^+\forall)$, (resp. $(\forall\exists/\exists^+\forall)$, $(\exists^+\exists/\forall\forall)$) denotes the class of languages that are accepted by a general (resp. perfect completeness, perfect soundness) two-move Arthur-Merlin proof system.

## 2. MODEL AND DEFINITIONS

We state and prove our main result for the Arthur Merlin games introduced by Babai [B]. Using the result of [GS] our main result applies also to the interactive proof systems of [GMR]. In this section we provide a precise definition of Arthur Merlin games and auxiliary terminology, in order to facilitate the presentation of our result. Since we are interested only in the complexity theoretic aspects of proof systems, we may assume that the prover (Merlin) uses an optimal strategy and therefore, with no loss of generality, is deterministic. In the following definition we assume that in all interactions of Arthur and Merlin, on inputs of the same length, the same number of messages are exchanged and that all these messages are of the same length. Clearly, this condition is immaterial and is only placed in order to facilitate the analysis.

**Definition 1 (Arthur Merlin games):**

An *Arthur-Merlin game* is a pair of interactive programs $\mathbf{A}$ and $\mathbf{M}$ and a predicate $\rho$ such that:

- On common input $x$, exactly $2q(|x|)$ messages of length $m(|x|)$ each are exchanged, where $q$ and $m$ are fixed polynomials and $|x|$ denotes the length of $x$.
- Arthur ($\mathbf{A}$) goes first, and at iteration $1 \leq i \leq q(|x|)$ chooses at random a string $r_i$ of length $m(|x|)$, with uniform probability distribution.
- Merlin's reply in the $i$-th iteration, denoted $y_i$, is a function of all the previous choices of Arthur and the common input $x$. More formally, $y_i = \mathbf{M}(x, r_1 \cdots r_i)$. In other words, $M$ is the strategy of Merlin.
- For every program $\mathbf{M}'$, a *conversation* between $\mathbf{A}$ and $\mathbf{M}'$ on input $x$ is a string $r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}$, where for every $1 \leq i \leq q(|x|)$ $y_i = \mathbf{M}'(x, r_1 \cdots r_i)$. We denote by $CONV_x^{\mathbf{M}'}$ the set of all conversations between $\mathbf{A}$ and $\mathbf{M}'$ on input $x$. Note that $|CONV_x^{\mathbf{M}'}| = 2^{q(|x|)m(|x|)}$.

- The predicate $\rho$ is a polynomial-time computable predicate. This predicate maps the input $x$ and a conversation $r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}$ to a Boolean value, called *the value of the conversation*. We associate $true$ with *accept* and $false$ with $reject$. The predicate $\rho$ is called the *value-of-the-game predicate.*

**Notation:** Let $\mathbf{A}$ and $\mathbf{M}'$ be programs and $\rho$ be a predicate as above.
Then $ACC_x^{\rho,\mathbf{M}'}$ denotes the set
$\{r_1 \cdots r_{q(|x|)} | \exists y_1 \cdots y_{q(|x|)} \text{ s.t. } r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)} \in CONV_x^{\mathbf{M}'} \;\; \& \; \rho(r_1 y_1 \cdots r_{q(|x|)} y_{q(|x|)}) = accept\}$.
Intuitively, $ACC_x^{\rho,\mathbf{M}'}$ is the set of all the random choices leading $\mathbf{A}$ to accept $x$, when interacting with $\mathbf{M}'$. Note that $ACC_x^{\rho,\mathbf{M}'}$ depends only on Merlin ($\mathbf{M}'$) and the predicate $\rho$, since we assume that Arthur follows the protocol. The ratio $\frac{|ACC_x^{\rho,\mathbf{M}'}|}{|CONV_x^{\mathbf{M}'}|}$ is the probability that Arthur accepts $x$ when interacting with $\mathbf{M}'$.

**Definition 2 (Arthur Merlin proof systems):** An *Arthur-Merlin proof system for language* $L$ is an Arthur-Merlin game satisfying the following two conditions:

- There exists a strategy for Merlin, $\mathbf{M}$, such that for all $x \in L$, $\frac{|ACC_x^{\rho,\mathbf{M}}|}{|CONV_x^{\mathbf{M}}|} \geq \frac{2}{3}$ . (This condition is hereafter referred to as *probabilistic-completeness.*)

- For every strategy $\mathbf{M}'$ and for any $x \notin L$, $\frac{|ACC_x^{\rho,\mathbf{M}'}|}{|CONV_x^{\mathbf{M}'}|} \leq \frac{1}{3}$ . (This condition is hereafter referred to as *probabilistic-soundness.*)

An equivalent definition is obtained by replacing $1/3$ by $2^{-p(|x|)}$ and $2/3$ by $1 - 2^{-p(|x|)}$, where $p(\cdot)$ is an arbitrary polynomial satisfying $p(n)\ 1$ ($\forall\, n\ 1$).

**Definition 3 (perfect completeness):** An Arthur-Merlin proof system with *perfect-completeness* for a language L is an Arthur-Merlin proof system for L satisfying:

$\forall x \in L \;\; |ACC_x^{\rho,\mathbf{M}}| = CONVSIZE$

Perfect-completeness, of an Arthur-Merlin proof system, means that Merlin **always** succeeds in convincing Arthur to accept inputs in the language.

**Definition 4 (perfect soundness):** An Arthur-Merlin proof system with *perfect-soundness* for a language L is an Arthur-Merlin proof system for L satisfying:

$\forall \mathbf{M}' \forall x \notin L \;\; ACC_x^{\rho,\mathbf{M}'} = \emptyset$

Perfect-soundness, of an Arthur-Merlin proof system, means that no matter what Merlin does Arthur **never** accepts an input not in the language.

## 3. ARTHUR MERLIN PROOF SYSTEMS WITH PERFECT COMPLETENESS

In this section we transform an Arthur-Merlin proof system to an Arthur-Merlin proof system with perfect completeness. This transformation preserves the number of interactions in the original Arthur-Merlin proof. The underlying technique is taken from Lautemann's proof that BPP is in the polynomial-time hierarchy [L]. (Lautemann's proof that BPP is in the polynomial-time hierarchy simplifies the original proof of Sipser [S].) The idea is to show that this technique works also for Arthur-Merlin proof systems. We think that this idea seems strange at first glance, trivial in second thought, but in fact is quite surprising and important. Lautemann's technique is commonly presented as a method of expressing a "random" quantifier by a universal and an existential quantifier. Suppose we are dealing with a subset, $W$, of $\{0,1\}^k$ and that this subset has cardinality either $\geq\ (1-\epsilon)\cdot 2^k$ or $\leq\ \epsilon\cdot 2^k$. The statement "most $r\in\{0,1\}^k$ are in $W$" can be substituted by the statement "$\exists s^{(1)}, s^{(2)}, ..., s^{(k)} \in \{0,1\}^k$ such that $\forall r \in \{0,1\}^k$ there $\exists i$ $(1 \leq i \leq k)$ such that $s^{(i)} \oplus r \in W$", where $s \oplus r$ is the bit-by-bit XOR of the strings $s$ and $r$. The strings $s^{(1)}...s^{(k)}$ are said to "*cover*" $W$. The statement "most $r \in \{0,1\}^k$ are not in $W$" can be substituted by the statement "$\forall s^{(1)}, s^{(2)}, ..., s^{(k)} \in \{0,1\}^k$ $\exists r \in \{0,1\}^k$ $\forall i$ $(1 \leq i \leq k)$ $s^{(i)} \oplus r \notin W$". Zachos showed that the above "simulation" can be used to swap quantifiers in a successive manner (for survey see [Z, Sch]). Zachos and Fuerer [ZF] then used this idea to show that bounded Arthur-Merlin proofs equal bounded Arthur-Merlin proofs with perfect completeness, by expressing the former proofs as a fixed quantifier sequence and applying a "swapping lemma" iteratively. For example, applying the swapping lemma to $(\exists^+\exists/\exists^+\forall)$ and using the BPP characterization [ZH] one gets $(\forall\exists^+\exists/\exists^+\forall\forall) = (\forall\exists/\exists^+\forall)$. Each such iteration is thus a straightforward application of the "simulation technique", and blows-up the size of the Arthur-Merlin game by an unbounded amount. Thus, this idea does not extend to unbounded Arthur-Merlin proofs. For our transformation it is necessary to extend the simulation technique to settings in which the witness set $W$ is not predetermined. In fact, in Arthur-Merlin games the set of random choices leading Arthur to accept is not defined, unless Merlin is specified. This fact is disturbing in the case that the input is not in the language and one has to guarantee that *no matter how Merlin acts* he cannot fool Arthur (except for low probability).

**An overview of the protocol** Without loss of generality, we assume that the error probability in the original Arthur-Merlin game is sufficiently small ( i.e. $\epsilon(|x|)\ \frac{1}{3q(|x|)m(|x|)}$ ). The transformed Arthur-Merlin game will consist of $k = q(|x|)m(|x|)$ original games played concurrently with related coin tosses, and Arthur will accept iff he accepts in one of these games. More specifically, Merlin starts the game by selecting carefully $k$ strings, $s^{(1)}, s^{(2)}, ..., s^{(k)} \in \{0,1\}^k$, and sending them to Arthur. These strings are selected to "cover" $ACC_x^{\rho,\mathbf{M}}$ in the case that $x$ is in the language. Arthur and Merlin now start to play $k$ copies of the original game. In round $j$, Arthur sends only one $m$-bit string $r_j$ and his move in the $i$-th game is defined as the bit-by-bit XOR of $r_j$ and the $j$-th segment in $s^{(i)}$ (i.e. Arthur's $j$-th move in the $i$-th copy is $r_j^{(i)} = r_j \oplus s_j^{(i)}$, where $s_j^{(i)}$ is the $j$-th $m$-bit block in $s^{(i)}$). Merlin answers by $k$ strings so that the $i$-th string equals the answer the original Merlin would have given in the $i$-th copy (i.e. the $i$-th $m$-bit block in Merlin's $j$-th message equals $\mathbf{M}(x, r_1^{(i)} r_2^{(i)} \cdots r_j^{(i)})$, where $\mathbf{M}$ is the original Merlin). Clearly, the perfect completeness condition is satisfied. It is less easy to see that probabilistic soundness is satisfied as well. Note that a cheating Merlin may select his answers for one copy of the game depending on his prospects in the other copies, and in particular arguing about $ACC_x^M$ is not sufficient.

Our argument, instead, consists of two claims: 1) the probability of winning the transformed game is bounded by the sum of the probabilities of winning each copy; and 2) the probability of winning a particular copy is bounded by the probability of winning the original game. (Trying to incorporate both claims in one counting argument leads to difficulties which are not encountered in Lautemann's original proof.)

**3.1. The Protocol** We denote the original Arthur by $\hat{\mathbf{A}}$, the original Merlin by $\hat{\mathbf{M}}$, and the original value-of-the-game predicate by $\hat{\rho}$. Let $\epsilon$ be the error probability, i.e. for $x \in L$ the Prob($\hat{\mathbf{A}}$ accepts) $> 1-\epsilon(|x|)$, and for $x \notin L$ the Prob($\hat{\mathbf{A}}$ accepts) $< \epsilon(|x|)$. On input of size $n$, $q(n)$ iterations are performed, at each iteration Arthur sends a message of length $m(n)$. When clear from the text we use $\epsilon, q, m$ for $\epsilon(n), q(n), m(n)$, respectively. Let $k = qm$. Without loss of generality we assume that $\epsilon < \frac{1}{3k}$. This can be achieved by performing sufficiently many copies of the original Arthur Merlin game in parallel, and ruling by the majority (see [B], [GS] and [BHZ]).

**Program for an honest Merlin:** Merlin's program consists of two stages. First, Merlin computes $k$ "sampling points" that are favorable to him, and sends them to Arthur. The second stage is a simulation of $k$ (related) copies of the original Arthur Merlin game.

*Preprocessing stage* Let $ACC$ be the set of random choices leading Arthur to accept in the original $\hat{\mathbf{A}}\hat{\mathbf{M}}$ game on input $x \in L$ (i.e. $ACC$ is a shorthand for $ACC_x^{\hat{\rho},\hat{\mathbf{M}}}$). Merlin selects $k$ strings $s^{(1)}, s^{(2)}, ..., s^{(k)} \in \{0,1\}^k$ so that for every $r \in \{0,1\}^k$ there exists an $i$ such that $s^{(i)} \oplus r \in ACC$. The preprocessing is said to have *failed*, if no such set of $s^{(i)}$'s exist. If the preprocessing does not fail then Merlin sends the $s^{(i)}$'s to Arthur. For sake of simplicity, we let Merlin send $k$ (arbitrary) strings (of length $k$-bit each) in case the preprocessing fails.

*Simulation stage* Merlin plays concurrently $k$ copies of the original game and computes Arthurs responses by XORing them with segments of the $s^{(i)}$'s. Each $s^{(i)}$ is partitioned into $q$ segments, of $m$ bits each, corresponding to the $q$ iteration of the original game. Namely, $s^{(i)} = s_1^{(i)} s_2^{(i)} \cdots s_q^{(i)}$, where $s_j^{(i)} \in \{0,1\}^m$. Formally, at each iteration $j (1 \leq j \leq q(n))$, Merlin preforms:

Receive $r_j$ For $i = 1$ to $k$ do begin $r_j^{(i)} \leftarrow s_j^{(i)} \oplus r_j$ $y_j^{(i)} \leftarrow \hat{\mathbf{M}}(x, r_1^{(i)} \cdots r_j^{(i)})$ End Send $y_j^{(1)}, y_j^{(2)}, ..., y_j^{(k)}$

**Arthur's program:** Arthur's program is identical to the original program of Arthur. Formally, for each iteration $j$ $(1 \leq j \leq q(n))$ Arthur performs:

Choose $r_j$ at random in $\{0,1\}^m$. Send $r_j$ Receive $y_j^{(1)}, y_j^{(2)}, ..., y_j^{(k)}$

**The value of a conversation**

Let $r_j^{(i)} = r_j \oplus s_j^{(i)}$, $y_j = y_j^{(1)} y_j^{(2)} \cdots y_j^{(k)}$, and $\bar{s} = s^{(1)} s^{(2)} \cdots s^{(k)}$. We denote by

$\rho_i(x, \bar{s}\, r_1 y_1 \cdots r_q y_q) = \hat{\rho}(x, r_1^{(i)} y_1^{(i)} \cdots r_q^{(i)} y_q^{(i)})$ the value of the $i$-th game. The predicate $\rho_i$ maps a conversations to 1 if and only if the conversation induced on the $i$-th copy of the original game is an accepting one. The value of a conversation is determined by the following polynomial-time predicate

$\rho(x, \bar{s}\, r_1 y_1 \cdots r_q y_q) = or_{i=1}^{k}\, \rho_i(x, \bar{s}\, r_1 y_1 \cdots r_q y_q)$

**3.2. The Perfect-Completeness of the protocol** We show that if the input $x$ is in $L$, then an honest Merlin (Merlin following the strategy outlined in subsection 3.1) always convinces Arthur. The argument is almost identical to the one in Lautemann (since $ACC$ is fixed!), and is given here for sake of self-containment (see also [ZH]).

**Lemma 1:** If $x \in L$ then the preprocessing does not fail.

**Proof:** We have to show that if $|ACC| = (1-\epsilon) \cdot 2^k$ and $\epsilon \leq \frac{1}{3k}$ then there exists a sequence, $\bar{s} = s^{(1)}, s^{(2)}, ..., s^{(k)}$ ($s^{(i)} \in \{0,1\}^k$), such that for every string $r \in \{0,1\}^k$ at least one of the $r \oplus s^{(i)}$ is in $ACC$. Furthermore, we will show that the statement holds for most sequences $\bar{s}$. We call a sequence $\bar{s} = s^{(1)}, s^{(2)}, ..., s^{(k)}$ *good* if for every $r \in \{0,1\}^k$ there exists an $i$ ($1 \leq i \leq k$) such that $r \oplus s^{(i)} \in ACC$. We consider the probability that a randomly selected sequence $\bar{s}$ is not good.

$$\begin{aligned}
Prob(\bar{s}\,\text{is not good}) &= Prob(\exists r\, \forall i:\ r \oplus s^{(i)} \notin ACC) \\
&\leq \sum_{r \in \{0,1\}^k} Prob(\forall i:\ r \oplus s^{(i)} \notin ACC) \\
&= 2^k \cdot Prob(\forall i:\ s^{(i)} \notin ACC) \\
&= 2^k \cdot \epsilon^k \\
&< (\frac{2}{3k})^k \\
&< 2^{-k}
\end{aligned}$$

The Lemma follows. □

**Lemma 2:** If $x \in L$ then Arthur always accepts.

**Proof:** By Lemma 1, Merlin can find $s^{(i)}$'s so that (when Merlin follows his program!) any sequence of choices made by Arthur leads to acceptance in at least one of the copies of the original game. The Lemma follows. □

**3.3. The Probabilistic Soundness of the protocol** We now show that for every input $x$ not in $L$, no matter what Merlin does, the probability that he convinces Arthur is less then 1/3. We consider the probabilities that Merlin $\mathbf{M}'$ leads Arthur to accept in the $i$-th copy of the original game. We first bound by $\epsilon$ the probability that $\mathbf{M}'$ leads Arthur to accept in the $i$-th copy of the original game (see Lemma 3). Hence, the probability that Merlin fools Arthur is bounded by $k \cdot \epsilon$ (Lemma 4). Let $\mathbf{M}'$ be any arbitrary program for Merlin. Recall that $ACC_x^{\hat{\rho},\mathbf{M}'}$ denotes the set of random choices leading Arthur ($\hat{\mathbf{A}}$) to accept in the original game (with game value predicate $\hat{\rho}$). We denote the set of random choices leading Arthur to accept in the $i$-th game of the transformed game by $ACC_x^{\rho_i,\mathbf{M}'}$. Namely, $r_1 r_2 \cdots r_q \in ACC_x^{\rho_i,\mathbf{M}'}$ if and only if $\rho_i(x, r_1\mathbf{M}'(x,r_1)\cdots r_q\mathbf{M}'(x,r_1\cdots r_q))$. Note that both $ACC_x^{\rho_i,\mathbf{M}'}$ and $ACC_x^{\hat{\rho},\mathbf{M}'}$ are subsets of $\{0,1\}^k$.

**Lemma 3:** Suppose that $x \notin L$. Then for every Merlin $\mathbf{M}'$ and for every $i$ $(1 \le i \le k)$

$$|ACC_x^{\rho_i,\mathbf{M}'}| \;\le\; \epsilon \cdot 2^k$$

**Proof:** The idea of the proof is that a Merlin which does well on a particular copy of the original game can be easily transformed into a Merlin which does (at least) as well in the original game. The transformed Merlin (which plays the original game) simulates the actions of the Merlin which plays $k$ games concurrently, using the real game as the $i$-th copy. A detailed proof follows. Assume, on the contrary to the statement of the lemma, that there exists an $\mathbf{M}'$ and an $i$, such that $|ACC_x^{\rho_i,\mathbf{M}'}|\ \epsilon\cdot 2^k$. We reach a contradiction by constructing a Merlin $\mathbf{M}''$, which does as well in the original game. First, $\mathbf{M}''$ runs $\mathbf{M}'$ on input $x$ to get the $k$ sample points $s^{(1)}, s^{(2)}, ..., s^{(k)}$ and saves $s^{(i)}$. Let $r_1, r_2, ..., r_j$ be the first $j$ messages that $\mathbf{M}''$ has received. To compute the $j$-th message, $\mathbf{M}''$ computes $r'_t = r_t \oplus s_t^{(i)}$ (for $1 \le t \le j$) and runs $\mathbf{M}'$ on input $x$ and $r'_1 r'_2 \cdots r'_j$ (i.e. $\mathbf{M}''(x, r_1\cdots r_j)$ is the $i$-th $m$-bit block of $\mathbf{M}'(x, r'_1\cdots r'_j)$). We now claim that

> **Claim**: $r \in ACC_x^{\rho_i,\mathbf{M}'}$ if and only if $r \oplus s^{(i)} \in ACC_x^{\hat{\rho},\mathbf{M}''}$.
>
> **Proof:** Suppose that $r_1\cdots r_q \in ACC^{\rho_i,\mathbf{M}'}$. Then $\rho_i(x, \bar{s} r_1 y_1 \cdots r_q y_q)$ is true, where $\bar{s} = s^{(1)}\cdots s^{(k)} = \mathbf{M}'(x)$ and $y_j = \mathbf{M}'(x, r_1\cdots r_j)$ $(\forall j)$. It follows that $\hat{\rho}(x, r_1^{(i)} y_1^{(i)} \cdots r_q^{(i)} y_q^{(i)}) = 1$, where $y_j^{(i)}$ is the $i$-th $m$-bit block in $y_j$, $s_j^{(i)}$ is the $j$-th $m$-bit block in $s^{(i)}$, and $r_j^{(i)} = r_j \oplus s_j^{(i)}$. Note that $y_j^{(i)} = \mathbf{M}''(x, r_1^{(i)}\cdots r_j^{(i)})$. Thus, $r_1^{(i)}\cdots r_q^{(i)} \in ACC_x^{\hat{\rho},\mathbf{M}''}$. Noting that $r_1^{(i)}\cdots r_q^{(i)} = r \oplus s^{(i)}$ one direction follows. The proof of the second direction is similar and the claim follows. $\Box$

By the above Claim, $|ACC_x^{\hat{\rho},\mathbf{M}''}| = |ACC_x^{\rho_i,\mathbf{M}'}|\ \epsilon \cdot 2^k$, which contradicts the hypothesis that the original game has error probability $\le \epsilon$. The lemma follows. $\Box$

**Remark:** A statement analogue to Lemma 3 is trivial in Lautemann's setting.

**Lemma 4:** Suppose that $x \notin L$. Then for every Merlin $\mathbf{M}'$ the probability that Arthur accepts is at most $k \cdot \epsilon$.

**Proof:** Clearly, for every Merlin $\mathbf{M}'$,

$$|ACC_x^{\rho,\mathbf{M}'}| = |\cup_{i=1}^{k} ACC_x^{\rho_i,\mathbf{M}'}| \le \sum_{i=1}^{k} |ACC_x^{\rho_i,\mathbf{M}'}|$$

Using Lemma 3, the statement follows. $\Box$

**3.4. Main Result** Using the equivalence of interactive proofs and Arthur Merlin proofs [GS], and combining Lemmas 2 and 4 we get

**Main Theorem (Theorem 5):** *If a language $L$ has an interactive proof system* (with $q(\cdot)$ iterations) *then $L$ has an (Arthur Merlin) interactive proof system with perfect completeness* (and $q(\cdot)+1$ iterations). $\square$

## 4. INTERACTIVE PROOF SYSTEMS WITH PERFECT SOUNDNESS

In the previous section, we showed that interactive proofs can be modified so that the verifier always accepts valid statements. What happens if we require that the verifier never accepts false statements? In this case we show that the set of languages recognized equals NP. The reader should note that the transformation of Goldwasser and Sipser [GS] does not preserve perfect completeness. Thus it is not clear that proving the above statement with respect to Arthur Merlin games yields the same result with respect to general interactive proofs. The difficulty can be resolved by modifying the transformation of [GS], using the approximate lower bound protocol of [GMS] (which has the perfect completeness property). We prefer to give a direct proof. The difference between interactive proofs and Arthur Merlin games is that in interactive proofs the verifier's $i$-th message $\alpha_i$ is a function of the input $x$, his random coin tosses $r$, and the previous messages of the prover (i.e. $\alpha_i = V(x, r, y_1 \cdots y_{i-1})$. After the last (say $q$-th) iteration, the verifier decides whether to accept or reject by evaluating the polynomial-time predicate $\rho(x, r, y_1 \cdots y_q) \in \{accept, reject\}$.

**Theorem 6:** *If a language $L$ has an interactive proof with perfect soundness then $L \in NP$*

**Proof:** Assume that for a language L, there exists an interactive proof with perfect soundness. Since the verifier is limited to probabilistic polynomial time, then for any input $x \in L$ there is a conversation that convinces him, and is of polynomial length. The NP machine guesses this conversation, checks that it is indeed a legitimate one and that it leads the verifier to accept. Namely, the machine guesses a random tape $r$ and a conversation $\alpha_1 y_1 \cdots \alpha_q y_q$, and checks that $\alpha_i = V(x, r, y_1 \cdots y_{i-1})$ (for every $i$) and that $\rho(x, r, y_1 \cdots y_q) = accept$. If $x \in L$ then, by the probabilistic completeness condition, there exist (many) accepting conversations. If $x \notin L$ then, by the perfect-soundness condition, there is no such conversation, and any guess of the machine will fail. $\square$

## 5. CONCLUDING REMARKS

Assuming the existence of secure encryption functions (in the sense of [GM]) and using the results of [GMW], one can easily demonstrate the existence of zero-knowledge interactive proofs with perfect completeness for every language in $IP$. Given $L \in IP$, first present an interactive proof with perfect completeness for $L$, and next apply the techniques in [GMW] observing that they preserve perfect completeness. However, it is not clear whether every language having a perfect (resp. almost perfect) zero-knowledge interactive proof (see [F] for definition) has a perfect (resp. almost perfect) zero-knowledge interactive proof with perfect completeness. Weaker statement can nevertheless be proven:

1) Every language having an interactive proof which is almost perfect zero-knowledge *with respect to the specified verifier* has an interactive proof with perfect completeness which is almost perfect zero-knowledge *with respect to the specified verifier* (again see [F] for definition).

2) Every language having an interactive proof which is almost perfect zero-knowledge and remains so under *parallel composition* (see [O] for definition) has an almost perfect zero-knowledge proof with perfect completeness.

The key observation in proving both statements is that almost all sequences $\bar{s}$ can serve as sampling points (see proof of Lemma 1), and thus having the prover randomly select and send a *good* $\bar{s}$ does not yield any knowledge. (In the simulation we use a randomly selected $\bar{s}$, which is most likely but not necessarily good.) Babai [B] showed that any Arthur Merlin game with a fixed number of interactions can be simulated by a game with two interactions. A similar proof applies to the hierarchy of interactive proofs with perfect completeness. Goldwasser and Sipser showed that the power of interactive proofs is not decreased when restricting the verifier to use only "public coins" [GS]. We have showed that the power of interactive proofs is not decreased when further restricting the system to have perfect completeness. *How else can interactive proofs be restricted without decreasing their power?*

**REFERENCES**


[A] Adleman, L., "Two Theorems on Random Polynomial Time", *Proc. 19th FOCS*, 1978, pp. 75-83.

[AGH] Aiello, W., S. Goldwasser, and J. Hastad, "On the Power of Interaction", *Proc. 27th FOCS*, 1986, pp. 368-379.

[B] Babai, L., "Trading Group Theory for Randomness", *Proc. 17th STOC*, 1985, pp. 421-429.

[BM] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Jour. on Computing*, Vol. 13, 1984, pp. 850-864.

[BHZ] Boppana, R., J. Hastad, and S. Zachos, "Does Co-NP Have Short Interactive Proofs?", *IPL*, 25, May 1987, pp. 127-132.

[F] Fortnow, L., "The Complexity of Perfect Zero-Knowledge", this volume.

[G] Gill, J., "Complexity of Probabilistic Turing Machines", *SIAM J. on Comp.*, Vol. 6, No. 4, 1977, pp. 675-695.

[GMS] Goldreich, O, Y. Mansour, and M. Sipser "Interactive Proof Systems: Provers that never Fail and Random Selection", *Proc. 28th FOCS*, 1987, pp.449-461.

[GMW] Goldreich O., S. Micali and A. Wigderson, "Proofs that yield Nothing But the Validity of the assertion and the a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, 1986, pp. 174-187.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GMR] Goldwasser, S., S. Micali and C. Rackoff, "The knowledge Complexity of Interactive Proof Systems", *Proc. 17th STOC*, 1985, pp. 291-304.

[GS] Goldwasser, S. and M. Sipser, "Private coins versus Public coins", this volume.

[L] Lautemann, C., "BPP and the Polynomial-time Hierarchy", *IPL*, 14, 1983, pp. 215-217.

[O] Oren, Y., "On the Cunning Power of Cheating Verifiers: Some Observations about Zero-Knowledge Proofs", *Proc. 28th FOCS*, 1987, pp. 462-471.

[Sch] Schoening, U., "Probabilistic Complexity Classes and Lowness", *Proc. 2nd Structure in Complexity Theory Conf.*, IEEE 1987, pp. 2-8.

[S] Sipser, M., "A Complexity Theoretic Approach to Randomness", *Proc. 15th STOC*, 1983, pp. 330-335.

[Z] Zachos, S., "Probabilistic Quantifiers, Adversaries, and Complexity Classes", *Proc. 1st Structure in Complexity Theory Conf.*, LNCS 223, Springer Verlag, 1986, pp. 383-400.

[ZF] Zachos, S., and M. Fuerer, "Probabilistic Quantifiers vs. Distrustful Adversaries", unpublished manuscript, August 1985. (see also FCT-TCS 1987.)

[ZH] Zachos, S. and H. Heller, "A Decisive Characterization of BPP", *Information and Control*, 69, 1986, pp.125-135.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on FOCS*, 1982, pp. 80-91.